Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

2021-06

PERSONALIZING TALENT DEVELOPMENT UTILIZING A NETWORK FRAMEWORK

Dietz, Andrew E.

Monterey, CA; Naval Postgraduate School

http://hdl.handle.net/10945/67700

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PERSONALIZING TALENT DEVELOPMENT UTILIZING A NETWORK FRAMEWORK

by

Andrew E. Dietz

June 2021

Thesis Advisor: Ralucca Gera
Second Reader: D'Marie E. Bartolf

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
PERSONALIZING TALENT DEVELOPMENT UTILIZING A NETWORK FRAMEWORK

5. FUNDING NUMBERS

6. AUTHOR(S) Andrew E. Dietz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The professional development and skill acquisition environment is complex. This environment is broad and chaotic with many talent development opportunities and training paths a job seeker could pursue to become qualified for a desired position; however, job seekers struggle to acquire the skills demanded by the jobs they are pursuing. Furthermore, job seekers also face the challenge of navigating through the various training opportunities, since it is hard to predict what training will best prepare them for the positions they want. Is there a better way to match job seekers to the skills they need to develop? This paper examines and builds upon existing network-based systems to develop a framework that represents the environment, identifies subsets of skills and training content, and builds personalized content training plans. By developing a framework that represents the environment and addresses the challenges job seekers face, users will benefit from meaningful training as a means of preparing for their next job.

14. SUBJECT TERMS
network science, recommender system, experiential learning model, CHUNK learning, graph database, skill development, personalized learning, talent management, professional development

15. NUMBER OF PAGES
97

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

PERSONALIZING TALENT DEVELOPMENT UTILIZING A NETWORK FRAMEWORK

Andrew E. Dietz
Major, United States Army
BS, University of Alabama, 2008
MA, Webster University, 2018

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by: Ralucca Gera
Advisor

D'Marie E. Bartolf
Second Reader

Wei Kang
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The professional development and skill acquisition environment is complex. This environment is broad and chaotic with many talent development opportunities and training paths a job seeker could pursue to become qualified for a desired position; however, job seekers struggle to acquire the skills demanded by the jobs they are pursuing. Furthermore, job seekers also face the challenge of navigating through the various training opportunities, since it is hard to predict what training will best prepare them for the positions they want. Is there a better way to match job seekers to the skills they need to develop? This paper examines and builds upon existing network-based systems to develop a framework that represents the environment, identifies subsets of skills and training content, and builds personalized content training plans. By developing a framework that represents the environment and addresses the challenges job seekers face, users will benefit from meaningful training as a means of preparing for their next job.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AC	Abstract Conceptualization
AE	Active Experimentation
BFS	Breadth-First Search
BLS	Bureau of Labor Statistics
CE	Concrete Experience
CHUNK	Curated Heuristic Using a Network of Knowledge
CHUNK-PD	Curated Heuristic Using a Network of Knowledge - Professional Development
CQL	Cypher Query Language
CTN	Comprehensive Training Network
DA	Department of the Army
DBMS	Database Management System
DOL	Department of Labor
ELM	Experiential Learning Model
GOOD	Graph-Oriented Object Database
GQL	Graph Query Language
GUI	Graphical User Interface
KLS	Kolb Learning Style
KLSI	Kolb Learning Style Inventory
KSA	Knowledge, Skills, and Abilities

LP	Linear Optimization Program
MBTI	Myers-Briggs Type Indicator
MOS	Military Occupational Specialty
NPS	Naval Postgraduate School
OECD	Organization for Economic Co-Operation and Development
ORSA	Operations Research Systems Analyst
RDMS	Relational Database Management System
RO	Reflexive Observation
SQL	Structured Query Language
STEM	Science Technology Engineering and Math
TDY	Temporary Duty

Acknowledgments

First and foremost, I cannot thank my advisor enough. Associate Provost for Graduate Education Dr. Gera Ralucca has championed every aspect of network-based personalized learning for NPS and associated academic institutions for years. She has guided the department and her students to discover the benefits that network science has to offer and the potential which has yet to be tapped. Leading by example and inspiring those around you are great qualities that are often over attributed, but Dr. Gera truly inspires her research teams.

I am deeply grateful for my second reader, D'Marie Bartolf. Without her assistance, I would still be lost in my own work without discernible direction or narrative.

Thank you both for your assistance at every stage of the research project.

I would also like to thank the entire CHUNK Learning and research team, specifically LTC(R) Michelle Isenhour and Erik Johnson, without whom this project would have never gotten off the ground. I would like to offer a special thanks to Applied Mathematics Department Chair Wei Kang for taking a chance on me, to Bard Mansager who was the best academic associate a master's student could ask for, and to CDR Clay Herring for representing the U.S. Navy in the finest tradition.

I next want to explicitly thank CPT Neal Mooers for the constant use of his whiteboard and his willingness to listen to me talk through ideas for hours at end. I also thank the entire Applied Mathematics class of 2021 here at Naval Postgraduate School (NPS) for pulling me through two years of the best education possible.

Finally, I want to thank my amazing wife. Without her unwavering support and dedication, I would have never achieved my accomplishments. Best Team Ever.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Job Training and Skills Development Environment

People are constantly looking for new or better employment. Many people seek jobs because they are underemployed; others seek new jobs because they want to advance their careers. In November 2020, there were 10.7 million unemployed Americans actively looking for employment [1]. The average time employed Americans stay in their current position is only 4.1 years [1]. And in a 2016 survey, over 50% of currently employed respondents were training for career advancement or were actively seeking a new job [2]. Job-seekers must work hard to get the job they want; it can take 100-200 applications to get a single job offer.

In order to get a good job offer, job seekers need the skills required for those jobs. Skills development is one of the most cost-effective strategies for securing meaningful employment [3]. For employers, finding job seekers who already possess the skills required for a job is one of the top factors in hiring decisions [4]. Most workers, however, admit they do not have the skills they need to be successful in their current jobs, let alone the skills needed for advancement [2]. The wide array of skill development options a job seeker could choose from is as vast as the number of trainable skills.

Job-seekers face the challenges of identifying the skills they need for the positions they want and finding training for those skills. There is an arbitrarily large (and always changing) number of potential jobs and skills associated with those jobs. Furthermore, for candidates, developing the skills desired by employers becomes more complex as the number of training methods to acquire new skills grows almost as quickly as the diversity of various attainable skills.

It is not enough for a job seeker to be highly educated or amass many skills when applying for a job. Job seekers must identify and acquire the right skills for their desired positions. The U.S. Department of Labor (DOL) Bureau of Labor Statistics (BLS) tracks thousands of job types in the U.S., and each job has its own set of education and training requirements [5]. While some intergovernmental agencies like the Organization for Economic Co-Operation

and Development (OECD) identify and track skill demands across dozens of skill areas [6], every employer has the opportunity to define unique job requirements for the positions they offer.

Even after a job seeker identifies the skills they require, they still must find a way to acquire those skills. The job training and skills development environment is chaotic. The potential combinations of jobs and ways to train continues to grow with every new job or training program introduced. Without a standardized means of training job seekers for the jobs they want, these job seekers are often left navigating the process on their own.

The key question this thesis seeks to answer is as follows: Can we develop a mathematical model to match job seekers to the skills they need for the jobs they want and the training content for those skills?

1.2 Research Goals

Our goal is to develop a mathematical model that represents the individual skills development environment and addresses the challenges job seekers face. This model should be able to identify the skills needed for any position and help job seekers find a combination of training to acquire those skills. We believe that this model should be able to accomplish the following four goals:

First, the model must be able to represent the vast range of potential positions and skills-based training content as a complex network while capturing the interconnected nature of the skills and training content. A complex network is well suited for modeling such interconnected relationships.

Second, the proposed model must be able to identify subsets of relevant skills and training content based on the job seeker's desired future job. As the collection of jobs, skills, and training content grows in the model, only skills and training content relevant to the desired position should be considered for analysis.

Third, the model must be able to evaluate the relevant training materials and build a personalized training plans for each job seeker. The model should incorporate the job seeker's personal learning style preferences and past experiences to minimize the training

requirements, allowing the job seeker to focus on only the training content that is most relevant to their desired position.

Finally, the model must be able to provide these training plans back to the job seeker in a format which is both flexible and engaging. These training plans should be provided in an interactive non-linear format that allows job-seekers to explore the training materials heuristically. Furthermore, these training plans should dynamically update as the job seeker completes content and acquires new skills.

1.3 General Approach

Our general research approach is to incorporate and build upon existing network-based systems in order to develop a mathematical model that addresses the above challenges in pursuit of the research goals. The three-part methodology we propose in Chapter 3 directly reflects the research goals introduced in Section 1.2.

First, we define a complex network that depicts the dominant influencing entities in the job training and skills development environment and the relationships between those entities. Using a complex network as the basis for our model enables path finding and analysis techniques.

Second, our model captures the most relevant data from the network and reforms it in a graph-based database framework. This method allows for efficient storage and access to the entire network of data while retaining the analytical properties of the original network.

Third, our model uses a recommender system to create a personalized training curriculum for each job-seeking system user. Our recommender system incorporates learning styles and content delivery modes based on specific desired positions for each job seeker. The recommender system then selects training materials which best align with their preferred learning style. Users who are presented with personalized curricula based on their learning style will benefit from a more engaging professional development and skill training process.

Our network-based model provides an interactive training plan to each user in the form of a personalized training network. By addressing the identified challenges and research goals, we provide a model to match job seekers to training content for their desired future positions.

1.4 Thesis Structure

This thesis is organized into five chapters: Introduction, Background, Methodology, Extensions and Future Research, and Conclusions. In Chapter 2, we introduce the central concepts of the research through an examination of relevant related works that most influence the methodology. In Chapter 3, we present our methodology for developing the proposed model that addresses the research goals described in Chapter 1. In Chapter 4, we introduce several model extensions that contribute to the model's utility and areas of future research beyond the scope of this work. Finally, in Chapter 5, we revisit the purpose of the research by recapping the benefits of the proposed framework.

CHAPTER 2:
Research Foundations

In this chapter, we introduce the central research concepts and a short background on existing research in these areas as they apply to this work. The field of network science forms the foundation for this thesis. Furthermore, this chapter also summarizes the theory of experiential learning, the Curated Heuristic Using a Network of Knowledge (CHUNK) learning model, and recommender systems since these concepts are at the core of our research methodology. Understanding recent relevant works in these topics is essential to developing our methodology, proposed in Chapter 3.

2.1 Application of Network Science

The concepts and terminology of network science are found throughout this research. The field of network science and the concepts derived from graph theory provide the basis for describing the interconnected nature of complex systems. The framework of our methodology, proposed in Chapter 3, is built upon network science and the work described below.

The practice of connecting related objects in a mathematical graph is not a new idea. The foundations of graph theory can be traced to at least the 18th century, when the famous mathematician Euler applied "geometry of position" to study the Bridges of Königsberg problem [7]. Today, we can provide a more formal set of graph theory concepts and used them to represent the concepts introduced in Chapter 1.

One of the principal goals of this research is to develop a mathematical model that represents job seekers, positions, skills, and training content as connected components. A *graph* (Definition 2.1.1) is the most logical mechanism for depicting these relationships. Graphs are commonly represented as a geometric diagrams on a two-dimensional plane. The elements of a graph are known as vertices. Vertices are represented as points on the plane. When two vertices have a defined relationship, a line (known as an edge) is drawn between to represent the relationship.

Definition 2.1.1 *Graph*

> *A graph G consists of a finite nonempty set V of objects called vertices (the singular is vertex) and a set E of 2-element subsets of V called edges. The sets V and E are the vertex set and edge set of G, respectively. So a graph G is a pair (actually an ordered pair) of two sets V and E. For this reason, some write $G = (V, E)$ [8].*

Graphs are often visual depictions of a larger, more complex *network*. The study of networks and the field of network science has emerged in recent decades. The definition of a *network* has taken many forms. Most definitions indicate that a network is collection of entities that are in some way related to one another along with additional information about the entities or their relationships. M.E.J. Newman, provides the follows succinct definition for a network:

Definition 2.1.2 *Network*

> *A set of items, that we will call vertices or sometimes nodes, with connections between them called edges [9].*

Newman observed that networks were abundant in everyday systems such as social, technological, and biological systems. In order to accurately represent complex systems, network structures and behaviors are controlled by data associated with vertices and edges. Network scientists must carefully design networks with clear definitions for the vertices and edges, and describe what data is associated with each entity. The quality of these definitions dictates the effectiveness of network science analytical methods [10].

The field of network science includes many analytical tools for measuring graph traversability in the form of shortest *path* (Definition 2.1.3) identification, vertex (or the network's average) *eccentricity* (Definition 2.1.4), and vertex *centrality* (Definition 2.1.5). With these definitions, we can compare and contrast similar graphs or subgraphs based on the distances between nodes and gain insights into which intermediate nodes are the most crucial to traversing the network.

Definition 2.1.3 *Geodesic Path*

> *A geodesic path is the shortest path through the network from one vertex to another. Note that there may be and often is more than one geodesic path between two vertices [9].*

In many connected networks, identifying the shortest path from one node to another among many possible paths is useful information. Examples include routes between cities along highways, or the path data takes across routers on the internet. In connected networks where paths between two vertices are all of the same length, path identification can be used to reveal sets of intermediary vertices along the paths.

Definition 2.1.4 *Eccentricity*

> *For a vertex v in a connected graph G, the eccentricity $e(v)$ of v is the distance between v and a vertex farthest from v in G [8].*

In connected networks *eccentricity* measures the "closeness" of vertices in the network. The min and max eccentricities of a graph correlate to the network's radius and diameter respectively. By comparing a network's radius, diameter and the average eccentricity, we develop a general sense of the network's dimensions. Networks with small diameters will have small geodesic path lengths, that may be useful in some types of network analysis or optimization.

Definition 2.1.5 *Betweenness Centrality*

> *The betweenness centrality of a vertex i is the number of geodesic paths between other vertices that run through i [9].*

$$x_i = \frac{\sum \frac{n_{st}^i}{g_{st}}}{n^2 - n + 1} \tag{2.1}$$

where:
n^i_{st} is the number of $s-t$ geodesics that i belongs to, and G_{st} is the total number of $s-t$ geodesics [11].

Betweenness centrality is a powerful analytic tool for identifying critical vertices in a complex network. This centrality measurement specifically examines graph traversability and geodesic paths. The more geodesics that include a vertex, the higher that vertex's betweenness centrality becomes. When examining a complex network with many similar vertices, the vertices with high centrality values are often considered to be more critical to the network since their removal would result in a higher average eccentricities across the network [12].

2.1.1 Network Properties

In addition to the common graph theory definitions in the previous section, Newman [9] describes a set of network properties. These network properties are used to evaluate network structure and behavior in order to differential different types of networks. He states that "the simplest useful model of a network is the random graph"; however, most networks in the real world do not behave like random graphs [9]. Newman proposes that in order to exploit network structures for analysis we need to examine common network features like clustering, the small-word effect, degree distribution, and communities.

The degree of **clustering** in a graph tells us how likely adjacent vertices are to have a common neighbor [10]. There are many forms of measuring the clustering in a network. One prominent method described by Newman measures the number of triangles and triplets in the graph [9]:

$$C = \frac{1}{\text{Size(N)}} \sum_i \frac{\#\text{ of } P_{3_i}}{\#\text{ of } T_{3_i}}, \forall i \in N \tag{2.2}$$

where:
N is the set of vertices, P_{3_i} is any path of length 3 where vertex i is the middle vertex, and T_{3_i} is any set of three vertices (including i) where each vertex is connected to the other two.

The **Small-World Effect** postulates that in most connected networks, any two vertices are connected by a relatively short path length [13], [9]. This holds true even when most vertices are not adjacent because of the formation of cliques or hubs (groups of vertices that are fully connected, or vertices with a relatively high number of adjacent vertices). We can measure the degree to which a network displays the small world property through the network coefficient ω:

$$\omega = \frac{L_r}{L} - \frac{C}{C_l} \tag{2.3}$$

where L is the average path length of the network, L_r is the average path length of a regular graph of the same size, C is the clustering coefficient of the network, and C_l is the clustering coefficient of an equivalent lattice network [13], [14].

The next network property that Newman describes is the **degree distribution** across the vertices of the network [9]. Degree distribution is measured as a histogram of the normalized probabilities that any particular vertex will have a degree k (the number of connected edges) [10], [9]. Measuring a network's degree distribution can help predict the likely degree of new vertices as the network grows.

Definition 2.1.6 *Degree*

The degree of a vertex v in a graph G is the number of edges incident with v and is denoted by $deg_G v$ or simply by $deg v$ if the graph G is clear from the context. Also, $deg v$ is the number of vertices adjacent to v [8].

Another network property that Newman examines is **communities**. Communities are densely connected groups within the network [9]. Communities within a network are useful in identifying groups of vertices with similar properties. There are many valid ways to measure and detect communities within a network. The community detection algorithm proposed by Girvan and Newman builds a hierarchical clustering tree of communities. The Girvan-Newman algorithm starts with one community for the whole network and progressively separates it into smaller communities until certain conditions exist [15]. The Girvan-Newman algorithm is shown in the example from Clauset, Moore, and Newman 2.1.

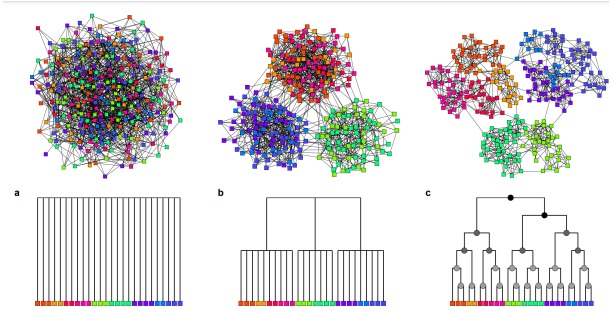


Figure 2.1. Girvan-Newman Clustering Example
Source: [16].

2.1.2 The Breadth-First Search Path Finding Algorithm

By nature, networks associate data and relationships directly onto network entities (vertices and edges) and not onto indexed tables of information. Without the use of hashable tables of data for reference, finding vertices that match specific properties requires the use of graph search algorithms. In their 1988 work, Ahuja, Magnanti, and Orlin identified four primary applications for search algorithms. They can find nodes that are reachable by another node via directed paths. They can find a set of nodes that can reach a specific node. They can identify the connected components of a network. And they can determine if a network is bipartite [17]. We are most concerned with the first application, finding sets of nodes that a reachable from a starting point node.

The Breadth-First Search (BFS) path finding algorithm shown in Figure 2.2 systematically builds a search tree of directed paths from a selected starting node. Every path in the search tree between the start node and any other connected node is a shortest path. The key feature of the BFS algorithm is that it considers the entire adjacency list of a particular node before moving on to the next node for path exploration [17]. Given a non-trivial graph, its adjacency matrix, and a start node s, the BFS algorithm is executed as follows:

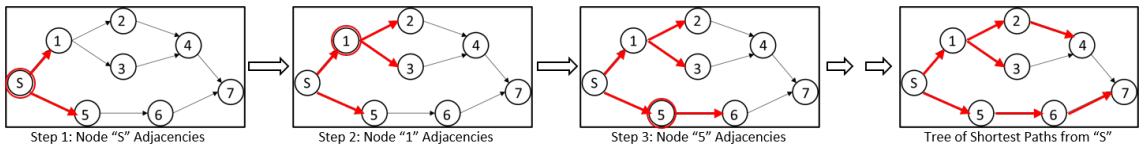


Figure 2.2. Example Breadth-First Search

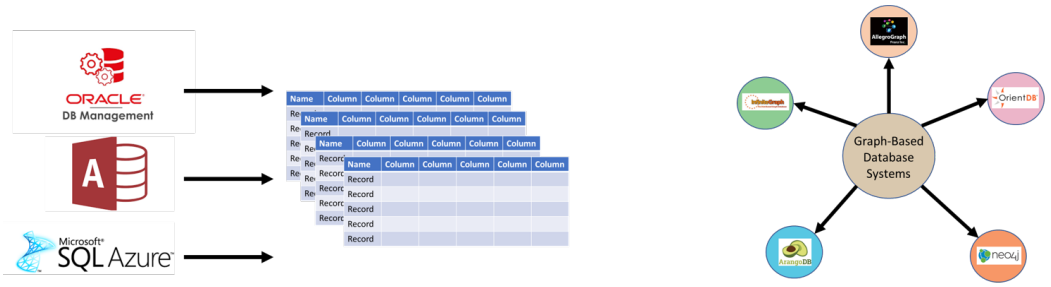
- Add start node s to a set of "reached" nodes
- Add start node s to a list of "nodes to explore"
- Iterate over any nodes in the list of "nodes to explore"; for each node:
  - Iterate over each adjacent node if they have not been "reached" yet:
    - add the adjacent node to the set of "reached" nodes
    - add the adjacent node to the list of "nodes to explore"
  - remove the parent node from the list of "nodes to explore"
- When no nodes remain in the list of "nodes to explore" the algorithm is finished

To promote computational efficiency, the basic BFS algorithm considers all edges between nodes to be the same length, and only retains the first path encounter to each node. Because of this, the BFS algorithm omits parallel paths of equivalent length in the resultant search tree.

2.1.3 Graph-Oriented Object Databases

One emerging application of network science is the Graph-Oriented Object Database (GOOD) in which data storage, representation, and manipulation is conducted via graphs [18]. Traditional Relational Database Management Systems (RDMSs) shown in Figure 2.3(a) are designed around relational tables with data stored in a structured format of rows and columns for efficient hashable data access [19]. These tables are then related to one another using sets of rules and accessed using Structured Query Language (SQL) based calls or transactions. The data storage schema of the RDMS designs, like those used by ORACLE or Microsoft Access, resides entirely on the data records (the rows in the relational tables) and not the relationship between the records.

The essential difference between a RDMS design and a GOOD design (implemented by Neo4j, AllegroGraph, OrientDB, and others as shown in Figure 2.3[b]) is that data is stored and transformed on the nodes and edges of graphs rather than on large data tables. The data properties associated with graph databases are accessed through Graph Query Language (GQL) calls and transactions. These queries access a combination of record caches as well as hashed relationship data [20]. In GOOD Database Management Systems (DBMSs), the focus of the data storage schema shifts from the records to the relationships within the system.



(a) Relational Database Systems

(b) Graph-Based Database Systems

Figure 2.3. Common Graph and Relational Database Systems

When the use case is appropriate, the performance of a GOOD system can outperform a RDMS design. This is predominantly due to the relational database's poor ability to process densely related data [21]. For example, a large table is a great way to store and access information on a bank accounts or historical record entries. This is because there can be many similar records, and rarely do they affect one another. However, if changes in any one bank account could affect changes in multiple others, a cascade effect could quickly overwhelm a system where each record in a large table has to be checked to see if its data changed. In a GOOD system, each vertex object maintains an internal record of adjacent entities (vertices) and the details of the relationship are stored directly on the edge between them. Storing information directly on a relationship means that it does not have to be explicitly computed or defined each time the relationship is called. Furthermore, this "index-free" method has the added advantage in that it can support complex networks with petabytes of data where no two entities have to be of a declared type (you do not need to define a table of bank accounts, and another table of bank transactions etc.). Even with a relatively small set of data (~500 nodes or records) a compound GQL query can be executed in Neo4j 15-30 times quicker than in a relational database [20].

Network science theory and tools are well suited for representing complex and interconnected systems such as the job training and skill develop environment. We can use complex networks to represent the interrelated entities associated with professional development. We could then use these networks to help job-seekers find the skills they need or training for those skills using network science techniques.

2.2 Learning Science and Experiential Learning

The field of learning science and the theory of experiential learning study how people acquire new knowledge and learn from experiences. Research works in the learning sciences have studied education and training theory and have established principles of personalized learning. The theory of Experiential Learning considers learning as a cyclic and highly individualized learning process. Together, learning science and experiential learning are crucial to establishing a personalized learning platform like the one implemented in CHUNK Learning that is described in Section 2.3. Understanding the principles and proper application of the learning sciences provides techniques for more meaningful professional development and skill training.

2.2.1 Learning Science

There are many similarities between job training and formalized education. In both cases an individual is developing skills that they can use later in life [3]. Traditional formalized education is oriented towards efficiently educating entire classes of students broad applications of knowledge for use in their adult lives or for a particular career field [22]. The principles of skill development found in the learning sciences can also be applied to the challenges associated with job training introduced in Chapter 1. The fundamental principles of learning science as they apply to skills development and job training are as follows. First, knowledge and skills are identified as collections of facts and procedures. Second, the goal of education or training is to transfer these facts and procedures to the student or trainee. Third, some elements of training or education are more difficult than others and should be taught or trained after the easier subjects. Finally, successful transfer of knowledge or skills can be tested to verify that they have been acquired. These principles are codified and implemented in the learning science style of *instructionism* (Definition 2.2.1).

Definition 2.2.1 *Instructionism*

> *The traditional vision of schooling prepares students for the industrialized economy by systematically indoctrinating students on facts and procedures of increasing complexity followed by testing [22].*

The practice of instructionism and similar styles have prevailed as baseline institutionalized education systems, even though alternative methods are available [22]. These widely employed institutionalized education systems are characterized by their attributes. They are designed for systematic instruction of entire classes of students en masse. They follow a centralized curriculum designed to teach as many students as possible as much as possible with a singular approach. The end goal of each system is to produce a degree that only establishes a relative baseline of knowledge that is expected to have been achieved [23]. While, effective, instructionism is impersonal, rigid, and general in nature. One alternative method which may be better suited for job training is *personalized learning* (Definition 2.2.2).

Definition 2.2.2 *Personalized Learning*

> *Personalized learning is roughly defined as the learning science concept in which the following conditions exist: 1) Accelerated systems of instruction with material tailored to each student based on their individual needs, skills, and interests; 2) a variety of rich learning experiences that collectively prepare students for success; 3) instructor or facilitators that are engaged in student instruction through managing the environment as well as expert guidance with the goal of encouraging the students to take responsibility for their own development [24].*

Personalized learning, where students are presented educational material tailored to their ability and learning style preference at a managed pace, has been shown to be effective [24]. In a national longitudinal study, students in personalized learning systems saw improvements in both math and reading scores well above the national averages [24]. Students thrive when they learn with dynamic educational materials that incorporate relevant engaging content [25]. One of the keys to making personalized learning successful is building systems where students are presented with engaging dynamic and cognitively challenging materials that personalized learning aspires to provide. These same principles could be applied to job training and skills development.

One approach to influencing others that can be applied to personalized learning, is Simon Sinek's "Golden Circle" (shown in Figure 2.4) [26]. In his "Why-How-What" approach,

Sinek emphasizes that leaders motivate others most effectively when they emphasize the purpose (the "why" component) behind the material they should learn or the new procedures they should adopt. Similarly, in learning science, personalized learning could be more effective if a student is motivated to learn a subject before being presented with the material. Once someone is motivated, Sinek proposes that a leader demonstrates how to use the material through relevant applications. Only after providing motivation and relevant application, the methodology (the "what" component) content of the material is presented to a follower (or student). In this manner, Sinek proposes that individuals will not only be more motivated to want to learn something new, but they will retain it and have the capacity to use it in broad applications long after training or testing is completed.

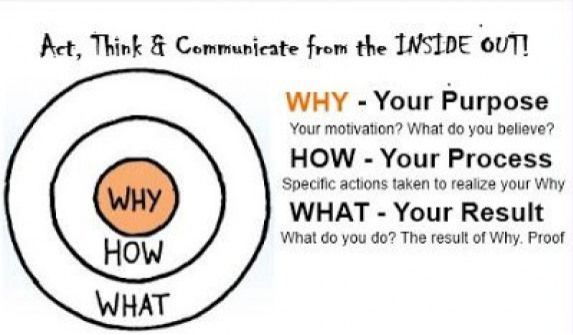


Figure 2.4. Simon Sinek's Golden Circle
Source: [26].

2.2.2 Experiential Learning

In his 1915 book, John Dewey introduced an alternative to traditional classroom rote-based education and suggested a practice of "learning by doing" [27]. After decades of study and refinement a formal definition of *experiential learning* was provided in a paper from Hoover and Whitehead [28]:

Definition 2.2.3 *Experiential Learning*

> *Experiential learning exists when a personally responsible participant(s) cognitively, affectively, and behaviorally processes knowledge, skills, and/or attitudes in a learning situation characterized by a high level of active involvement [28].*

Experiential Learning Theory Model

Experiential Learning Theory recognizes that learning is a process not a measured outcome. As a process, learning is dynamic and transformative based on the learner's experiences. One pioneer in the field of Experiential Learning, David Kolb, developed the prominent Experiential Learning Model (ELM) commonly referenced today [29]. In his model, Kolb portrays experiential learning as a cyclic process (shown in Figure 2.5) where learning never truly finishes. Experiential Learning begins with each new concrete experience. These experiences are observed and conceptualized. Eventually, we experiment with what was observed and conceptualized, and this results in new concrete experiences.

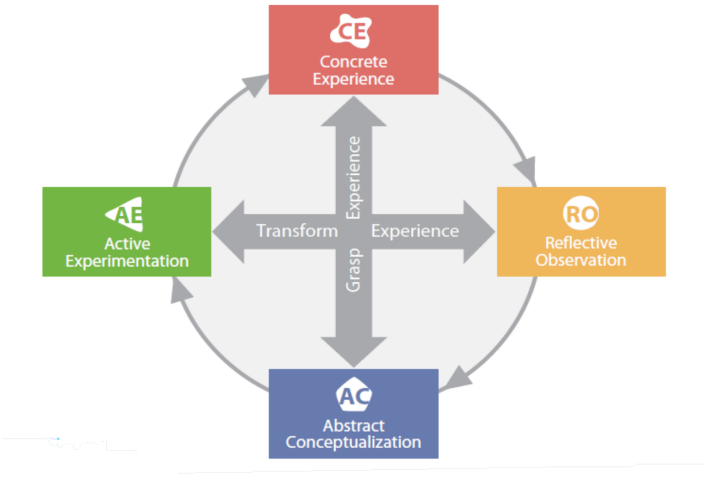


Figure 2.5. The Experiential Learning Cycle
Source: [30].

According to Kolb: "knowledge results from the combination of grasping and transforming experience" [29]. Experiences are grasped through the diametrically opposed learning modes of Concrete Experience (CE) and Abstract Conceptualization (AC). Experiences are transformed through the diametrically opposed learning modes of Reflexive Observation (RO) and Active Experimentation (AE) [30]. Kolb's ELM proposes that learning does not solely exist in any one of these leaning modes, but emerges as a result of an experience that combines some elements of each of the four learning modes. He proposes that knowledge acquired through this more holistic cycle is more well-rounded since it engages different sectors of the human brain simultaneously.

Kolb Learning Style

In addition to the ELM learning modes, Kolb proclaims that individuals have unique propensities and preferences that are defined as learning styles [31]. These learning styles are far more than just personality types, as might be predicted through a Myers-Briggs Type Indicator (MBTI) battery; they also relate to which learning modes are most compatible or engaging for the individual. While the original ELM defined and studied four primary Kolb Learning Styles (KLSs): Accommodating, Diverging, Converging, and Assimilating - these have been refined into the nine learning styles show in Figure 2.6 to better account for the distinctive preferences observed in each style.

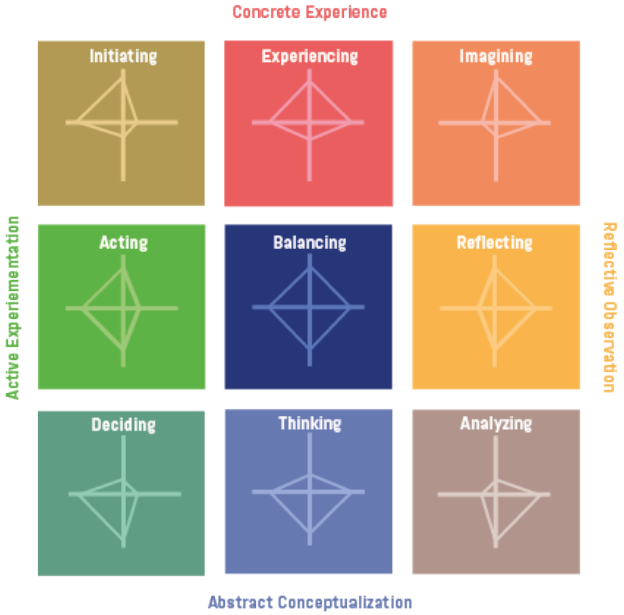


Figure 2.6. The Nine Learning Styles
Source: [30].

Each of the nine KLSs can be graphically associated to a region on the graph depicted in Figure 2.6. We define the horizontal axis as a gradient of how someone "transforms experiences" on a scale from AE to RO, and vertical axis as a gradient scale of how someone "grasps experiences" from AC to CE. Thus, with a defined Euclidean plane we can now plot regions that roughly represent each of the nine KLSs [30]. Consequently, someone who prescribes to the *Initiating* style prefers to grasp experiences through CEs and prefers to transform those same experiences through AEs. Initiators may prefer web based or hands-on experiences grounded in real life situations. Someone with a *Thinking*

KLS prefers to grasp experiences through ACs and equally prefers to transform experiences through AEs as well as through ROs. A thinker prefers individualized working environments unlike an Initiator, and may prefer studying a text over actively experimenting with new subjects or materials [32], [31].

Application of the ELM in a classroom setting for specific KLSs requires deliberate and personalized experiences. Application of the ELM also requires upfront understanding of the educator's role in the learning cycle [30]. Blending the educator's role and the learning mode maximizes the success of experiential learning. Between CE and RO on the ELM cycle, learning should be facilitated with learner and meaning focused inside-out-based experiences. Between RO and AC, students should analyze and organize new subject materials, such as with lectures or texts from experts. Between the AC and AE modes, learning is results oriented through objective evaluation. In the last quadrant between the AE and CE learners apply knowledge through collaborative or interactive coaching [30], [32].

Every learner has their own preferred KLS. Kolb defines these learning styles to help describe the unique way learners progress through the learning cycle and their learning mode preferences [30]. While it is important to note that a KLS is not a fixed psychological trait, identifying a learner's preferences creates a basis for personalized learning. Identifying a KLS is preferably done using a standardized battery called the Kolb Learning Style Inventory (KLSI). In the KLSI survey, respondents are given a series of incomplete sentences and asked to rank order phrases that complete the sentences. In this "Forced-choice format" the KLSI quantifies an individual's biases toward the four learning modes. These quantified results are then interpreted graphically on the plane behind Figure 2.6 to describe the likely learning style preference. Once one of the nine KLSs has been identified, the learner can use this to gain insight on their own learning style, or it can form the foundation for a personalized learning plan.

2.3 CHUNK Learning

One recently implemented personalized learning system is the CHUNK Learning network [33]. In the CHUNK model, students access a web-based network of academic content through their unique user profile. The CHUNK system is designed to build an interactive network of personalized content for each user. The CHUNK system relies on concept map-

ping a library of academic content into modularized elements based on the content contained in the element. The system capitalizes on this content library by presenting an interactive view of curated content to the user as seen in Figure 2.7.

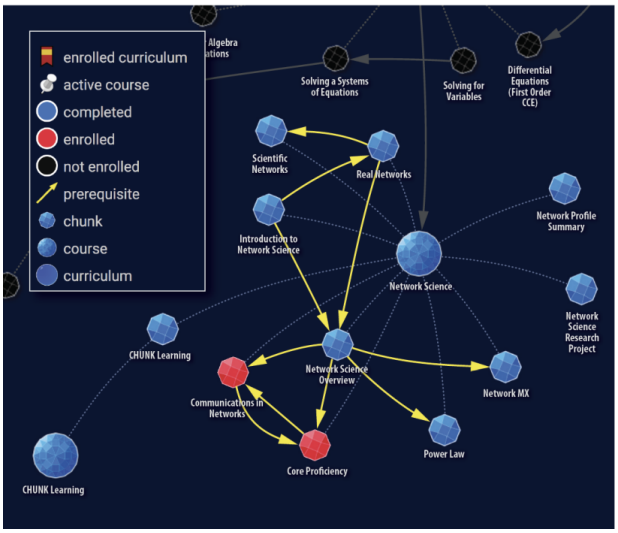


Figure 2.7. Curated Heuristic Using a Network of Knowledge
Adapted from [33].

The CHUNK Learning model is designed as an interactive personalized learning framework that presents educational content to students in an academic environment [34]. Two significant requirements identified in the CHUNK model are establishing the student profiles and managing the library of educational content. Student profiles are established every time a new user enrolls in the system. Each profile requires the student to manually determine their influence factors by identifying their interests as keywords from a large, finite, and minimally defined set of topics. Each element in the library of content is similarly keyword tagged from the same set of keywords.

This metadata assignment enables the CHUNK system to match content of interest to the students and build relationships between similar content within a hierarchical structure that mirrors the structure seen in traditional classroom education. The content hierarchy used in the CHUNK Learning System is codified in Cleven's 2019 work [34]. The content hierarchy has evolved to the one now seen by CHUNK Learning users in Table 2.1. The ability to separate major topics into individual training artifacts while retaining cognitive associations in a hierarchy could be applied to both academic and professional development content.

Table 2.1. CHUNK Learning.net Hierarchy

Adapted from [35].

Module Type	Description
Activity	A chunklet is a coherent collection of one or more Activities. The smallest subdivision of educational material available in chunklearning.net is an activity.
Chunklet	A chunk consists of several chunklets arranged in a conventional structure usually consisting of four types: why, how/what, methodology, and assessment. chunklets form the basic building blocks in the assembly of a chunk.
Chunk	Within the chunklearning.net system, course content is divided into several modules, commonly referred to as chunks.
Unit	An intermediary level between topics and chunks; a unit consists of several chunks, and is similar to the major units of an academic course.
Topic	Roughly equivalent to an academic course, a topic is a collection of units required to obtain a certification or course credit.

This hierarchy retains the original structure while adding an intermediary tier and introducing the possibility for sub-modules to have multiple parent nodes. This added flexibility allows a student to observe a lecture or complete an activity for one chunklet while receiving credit for a related chunklet [35].

Each chunk module is designed to present chunklets arranged according to a "Why-How-Methodology-Assessment" template (as shown in Figure 2.8) that mirrors Sinek's Golden Circle in Figure 2.4 [33], [26]. Users are first introduced to the purpose behind learning a topic; then they are exposed to practical real life applications of the new topic. After establishing why a topic is important and how it can be used in real life, chunklets teach a user the methodology behind the why and how. These "what" modules emphasize the learning outcomes and the new information required for a student to be able to eventually apply the knowledge themselves. In the fourth ('Assessment') category, students apply their knowledge through projects or evaluations.

The CHUNK model can be especially appealing to adult learners who have already acquired a broad set of skills through life experience or who desire a flexible self-managed format. This model based on short focused academic modules allows a student to manage their own

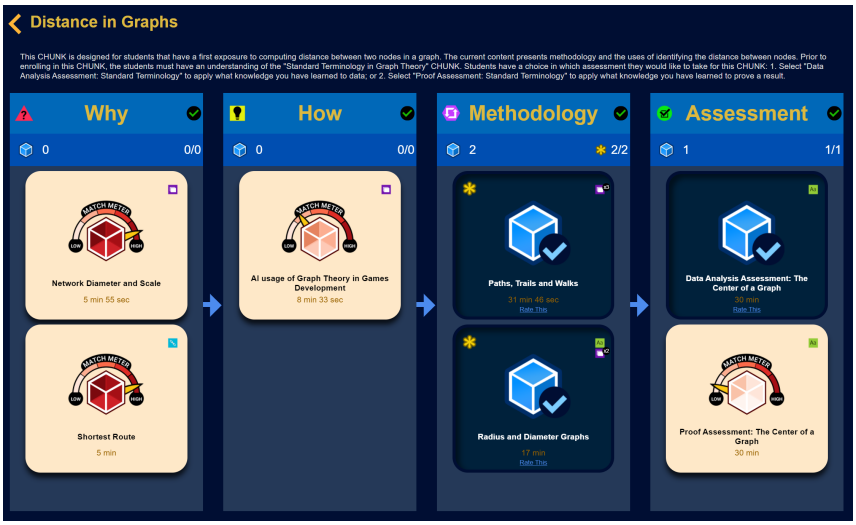


Figure 2.8. Why-How-Methodology-Assessment Format Inside Each Chunk
Source: [33].

pace through the coursework. Since the CHUNK model produces a curated personalized learning framework for each user to explore (as described in Section 2.2), students can actively explore content to meet their own learning goals [33].

2.4 Recommender Systems

Recommender systems (Definition 2.4.1) can help identify specific selections when a user is unfamiliar with the choices or the problem at hand. In Section 1.1 we introduce the job seeker's challenge of finding training for specific job required skills in a chaotic environment. Some of the reasons may be the complexity involved in picking training, the sheer volume of training options, or the difficulty in finding access to the training. In this section, we introduce some recommender system techniques and ways to evaluate what make a good recommender system. In Chapter 3, we develop a recommender system to help job-seekers find the right set of training content to acquire the skills they need for their next job.

Definition 2.4.1 (Recommender Systems) *Recommender Systems*

> *Recommender systems are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user... The*

suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read [36].

2.4.1 Recommender System Techniques

Recommender systems can apply any variety of approach when selecting items to recommend. Most recommender systems employ some degree of customization based on the system needs and data constraints. In their book, Ricci et al. [36] describe four recommender system techniques that generally describe they style of data inputs and outputs.

The first technique Ricci et al. describe is **collaborative filtering**. This method relies on high quality explicit feedback, such as a one-to-five-star user rating, as the primary input. Collaborative filtering recommender system produce recommendations through specific techniques like neighborhoods of similar items or by latent matrix factorization models such as single value decomposition. Collaborative filtering accuracy can be improved with the inclusion of implicit feedback, such as user behavior, but this further increases the computational cost for diminished results [36]. Some drawbacks of collaborative filtering include the lack of diversity and decreased ordering accuracy of results when compared to other algorithms. Additionally, collaborative filtering techniques cannot provide recommendation until sufficient ratings exists for an entity [37].

Content-based recommender systems make relevancy evaluations based on the internal metadata and content of the entities. This is in contrast to using community provided metadata as the primary recommender system input seen in collaborative filtering techniques [36]. In his 2017 work, Benzi [37] employs a content-based recommender system to build song playlists using features extracted direct from the song metadata. Instead of using ratings that would promote songs and artists which were already popular, he extracts features directly from the music to discover and recommend new songs or playlists. Content-based algorithms do increase recommendation heterogeneity but are computationally expensive since they require the extraction and analysis of content metadata.

One approach to content-based recommender systems is through **content maps**. Similar to Benzi's approach to selecting songs for a playlist, content-maps extract information about the items for evaluation. However, rather than directly comparing content for every item,

content-mapping builds relationships among the items and these relationships dictate which items the recommender system chooses from. The Metis Recommender System proposed by Underwood et al. identifies the Knowledge, Skills, and Abilities (KSA) that a student possesses and associated academic activities, and uses the content map to determine which activities to choose from [38]. This proposed content-map based recommender system seeks to improve learner interest and achievement by only presenting new activities that the student is ready for. A similar approach could be taken with job skills and training content.

For many systems, recommendations based on content or collaborative filtering is insufficient. There may be additional considerations such as user demands, item properties, or metadata age that should affect the recommendation outputs. **Constraint-based recommender systems** include bi-directional evaluations between the user and the items when computing recommendations [36]. This knowledge-based technique is useful for filtering data prior to computation. However, constraint-based recommender systems require extensive data engineering, for each item considered, in order to access and incorporate the necessary metadata.

Another approach to recommender systems takes into account the larger environment beyond the user and the item(s) recommended. These **context-aware recommender systems** consider external indirect influences like time, space, or even other recommender system users when computing recommendations. For job seekers pursing skill training options, a context-aware system would consider training availability dates or if 15 other job seekers have registered for a five-person training class. Normally, the recommender system may only consider the training modules and the user, but a context-aware recommender system adjusts the computed results accordingly.

2.4.2 Improving Recommender Systems

Many effective recommender systems are more than a direct implementation of one of the techniques described in the previous section. Benzi concludes that "the best recommender systems can only come from the merging of both content-based and collaborative filtering models" [37]. Benzi is describing the practice of **hybrid** recommender systems. A hybrid recommender system is any combination of two or more of the primary techniques. By combining multiple techniques, a single recommender system can improve its effectiveness

while avoiding some of the weaknesses that are associated with specific techniques.

A good recommender system should be capable of more than accurate prediction of which item(s) a user likes best. Effective recommender systems also provide accurate discovery of new material that a user has no prior opinion of [36]. Predictive recommender systems are difficult to develop since their effectiveness cannot be measured until after then system is implemented and users have the opportunity to provide explicit feedback. A fuzzy tree hybrid (part collaborative filtering and part content-based) recommender system proposed by Wu et al. incorporates user profiles to improve the accuracy of e-learning predictive recommendations [39]. Wu et al. discovered that learning activities and learner profiles often present tree-like structures. They conclude that a good recommender system will incorporate rather than fight this feature. The inclusion of user profiles allows the recommender system to compare and evaluate the user and the content without the user having prior opinions on the e-learning content.

Recommender systems are frequently used with very large data sets; sometimes the size of the data available is too large for efficient computation or provides unnecessary detail. Ricci states that "due to the sheer size of the data sets, aggregation becomes necessary to help summarize the extent to which an item satisfies a user's preferences" [36]. As the size of the data sets grows the strain on the recommender system also grows. To compensate for the increased computations **aggregation functions** are introduced. Aggregation functions combine characteristics of related items to provide an aggregated value that is then compared to other aggregated values for recommendations [36]. One example might be aggregating the songs of an album prior to recommending albums to a user.

CHAPTER 3:
Methodology

In this chapter, we propose the fundamental methodology for addressing the research problem. In Section 3.1, we elaborate on the research problem, applying context from related works to the problem, the four major challenges identified in Chapter 1, and how this work mitigates each of them. In Section 3.2, we provide a general overview of the framework for the research methodology. In Section 3.3, we define a comprehensive network in which the research model resides and the underlying foundations of the proposed model. In Section 3.4, we propose a method for forming the comprehensive network as a graph-based database that facilitates interaction between the training network data and the recommender system. In Section 3.5, we describe the recommender system at the core of the proposed model and the algorithms the recommender system uses to provide optimized content to the users. Collectively, this new framework forms a new environment we call the Curated Heuristic Using a Network of Knowledge - Professional Development (CHUNK-PD) environment. The CHUNK-PD environment provides a means for job-seekers to acquire skills for a targeted position, while the CHUNK Learning framework provides a means for students to acquire academic knowledge for a specified curriculum.

3.1 Problem Statement Re-examined

Job seekers face a chaotic job training and skills development environment. In Chapter 1, we identified two significant challenges they face–identifying the skills they need to acquire for the jobs they want and finding the training materials to acquire those skills.

Interconnected systems are easier to understand and navigate through the use of representative systems like complex networks discussed in Section 2.1.1. By modeling the job training and skills development environment as a complex network we can explicitly define the network's entities and data. As a complex network, we can apply network science techniques (such as search algorithms seen in Section 2.1.2) to identify skills for job seekers.

Matching users to training content is challenging due to the depth and variety of skills and training content. Organizing training material into a hierarchical structure breaks down

training content into well-categorized manageable modules. Training content should be categorized and organized like the academic material structure employed by the CHUNK system from Section 2.3.

Storing the network data in a graph-based database like those introduced in Section 2.1.3 would facilitate efficient network interaction while retaining network properties and the use of network-based analytical tools. Furthermore, we can categorize the subjective content styles and learning preferences of the users using the framework provided by experiential learning from Section 2.2.2. This categorization of abstract preferences can provide a basis for mathematical comparison to the training content.

Ultimately, our mathematical model needs to employ a recommender system like those introduced in Section 2.4 to optimize the training material modules that would go into each job seeker's training plan. Hybrid recommender systems that incorporate collaborative community provided data and data extracted from content while being aware of the environmental context provides a well-rounded approach to presenting the most relevant training content to each user.

3.2 Overview of Proposed Methodology

The general approach proposed in this thesis is to develop a mathematical framework based on complex network modeling to address the challenges described in Section 3.1. We apply this new framework to an interactive system (depicted in Figure 3.1). This framework requires a job-seeker and their desired position as system inputs, then produces a map of a personalized training curricula for the user as an output. This framework is named the CHUNK-PD environment.

Each curriculum presents an optimized set of training content for a desired position. This approach captures and organizes relevant information into the framework of a comprehensive training network, then processes this data through a recommender system that produces unique personalized training network for each user/position combination.

The approach described in Figure 3.1 builds on the fundamentals of graph theory to define and analyze a comprehensive network that represents relevant data as well as the dynamic relationships within the data sets. A graph-based object-oriented database allows for ef-

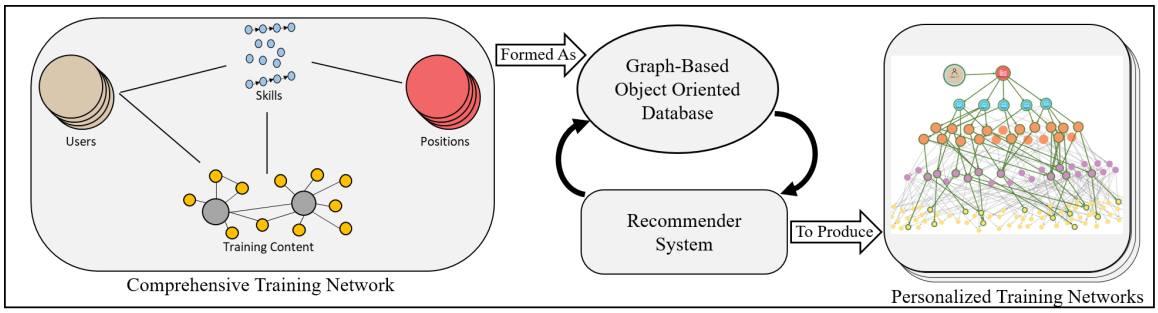


Figure 3.1. Skill Training System Model Overview (green edges identify the user's personalized network)

fective storage and recall of information directly on the relationships between the vertices of the network. The recommender system uses path identification and linear programming optimization in sequence to identify the best combination of training content for each user. Ultimately, the model's desired output is to produce individualized training plans (visualized as networks) that reinforce the non-linear and personalized learning approach to training and education stressed in this thesis.

We now present the details for each of the three steps in Figure 3.1.

3.3 Step 1: The Comprehensive Training Network

Before we can build personalized professional development training plans for CHUNK-PD users, we need to describe and define the environment job-seekers interact with. We define a Comprehensive Training Network (CTN) to organize and represent the elements and data associated with the job training and skills development environment introduce in Section 1.1. The network shown in Figure 3.2 contains four primary node sets: users, positions, skills & attributes, and training content. Network edges initially exist in four distinct edge sets. First, edges connect users and the training content they have already completed. Next, edges connect users and skills they have acquired or trained. Third, edges connect positions and the skills required by each individual position. The fourth edge set is a mapping of skills to the content that trains the skill. We explain the subgraphs of this network and its functions in greater detail in the following sections. Once we define a network which represent the environment we can describe how to interact with the information available in the network.

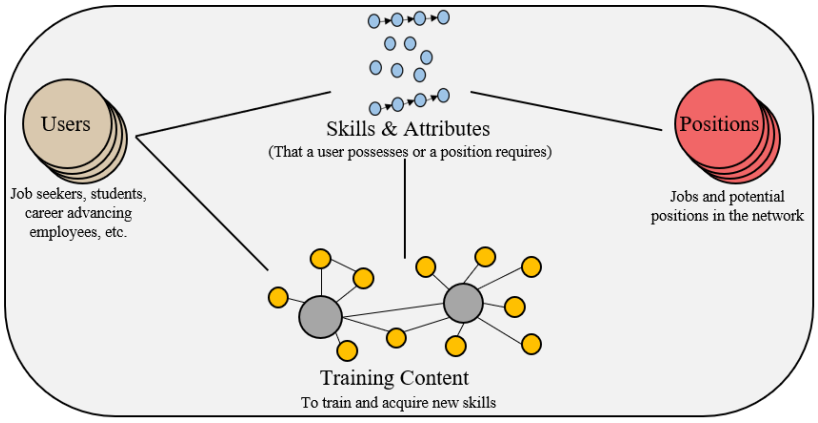


Figure 3.2. Comprehensive Training Network (CTN) Diagram

3.3.1 The Four Primary Network Vertex Sets

We will now introduce the network formally. The four primary vertex sets the model is based upon are defined here.

Users

The system's *users* are potential job seekers. They could be academic students looking to eventually enter the job market, active job seekers looking for immediate employment, or current employed professionals seeking to advance in their career path. Each user is represented as a vertex in the network and is related to the skill & attributes vertices, as well as training content vertices, through one of two edge types defined in Section 3.3.2.

Positions

The system's *positions* are potential jobs in the network a user could apply to (or hold). The position vertices are related to a set of skills or attributes that the position requires. These edges connect the different elements of the job description or the job requirements to each position. Unlike the edge set of skills related to users, the relationships between positions and skills do not change over time based on user training or changes in network size.

Skills

The system's *skills* represent the talents, skills, and attributes that a user has acquired through job training and experience; they are also the stated job requirements specified

by an employer for a position in the network. The conglomeration of job requirements and employee attributes across the vertex set of positions could become unwieldy. Therefore, the collective set of requirements and attributes is clustered by likeness into a manageable set of skills and attributes. This vertex set is sparsely connected as some skills are multi-leveled while others are stand-alone skills or attributes associated with positions in the network.

Training Content

The final set of vertices in the network represents *training content* modules (described in Section 3.3.3). The subgraph topology of the training content is similar to the CHUNK Learning model introduced in Section 2.3. The subgraph of training content forms a hierarchical subgraph consisting of units, chunks, chunklets, and activities. There are many edges between vertices in this segment of the network as some content is relevant to multiple topics, and some topics could be considered as a pre-requisite to other content, depicted by oriented arcs.

For context, we acknowledge that the CTN may be expanded beyond these four node sets to represent more elements forming a multi-layer network. Some of these elements could include additional actors (such as facilitators, coaches, hiring agents, etc.), temporal elements (such as job opening windows, or training content availability), or spacial constraints (such as job/user locations or training sites). These are addressed as framework extensions in Chapter 4.

3.3.2 Users, Positions, and the Skills They Share

Expanding on the definitions introduced in Section 3.3.1, this section explores the properties of the user and position vertices as well as the indirect relationships between them. In the proposed network model, user and position vertices are indirectly related through the skill vertices as shown in Figure 3.3. Users are related to those skills they have already acquired. Likewise, position vertices are related to the skills required or implied by the job description. The implication this research seeks to capitalize on is that: once a user has acquired every skill required by a position, then that job seeker is now fully qualified for that position.

In Figure 3.3 the users are related to acquired skills that the user inherently possesses through current employment, education, on-the-job training, or by having completed sufficient train-

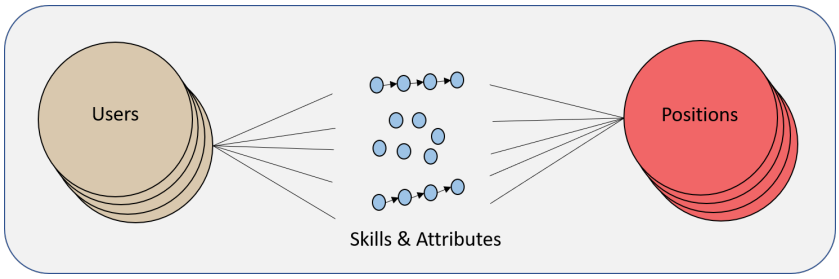


Figure 3.3. CTN Subgraph of Skills Associated with Users and Positions

ing content. Since job qualification is ultimately the responsibility of the applicant, for our work the system allows each user to self-identify any of their pre-existing skills or attributes. Since the system will produce personalized training plans based on users' existing knowledge and needed training, users are not presumed to be of any specific background area or level of expertise. As such, the system does not assume that any particular user is tied to a single desired position or fits into any ill-defined category of user type. Therefore, the model needs to allow for each user to be represented as a unique user node, that will be presented with a personalized training network.

In contrast to other existing methods that build elaborate models of user attributes (as seen in Section 3.5), the core of our user's profile is a single $1x4$ vector (annotated as κ ("kappa")) that describes the user's KLS as descried by Kolb [30]. Each element of κ represents the users bias towards each of the ELM's four modes of learning (on a scale of zero to one). We describe κ as follows:

$$\kappa = [\text{CE, RO, AC, AE}], \quad \text{where} \quad \text{CE, RO, AC, AE} \in (0, 1].$$

We use Kolb's model for learning styles since they are well defined established through significant research. Furthermore, the KLS for a new user can be established in several ways: through a formal KLSI survey, by "hand-picking" one of the nine established types in the KLSI 4.0 guide [30], by customizing a location on the plane in Figure 2.6, or through an interpretation of an MBTI evaluation. The results of any of these methods can be translated into a user's κ vector. Each KLS can be translated on to the two related AE $\leftrightarrow$ RO and CE $\leftrightarrow$ AC axes in Figure 2.6 (each axis on a scale from -1 to $+1$). If a user's RO - AE value

is mostly AE the plotted location will be closer to −1 on the RO-AE axis. If the same user's CE-AE score is mostly AC the plotted location will be close to −1 on the CE-AC axis.

The position vertices in the subgraph of the CTN shown in Figure 3.3 each represent a single employment position in an unbounded marketplace. In addition to identifying metadata such as job title or unique ID number, each position is related to the skill vertices required by the job description. These related skills can be singular related job requirements (such as "has a Commercial Driver's License") or they could be tiered skills (such as "Python programming: beginner" or "data processing: expert").

While a user can be added to the system with skills not represented in the network, any position added to the network must have a complete set of related skill vertices. An example of the skills that may be associated with a specific position is shown in Figure 3.4. This subgraph depicts the GS1515-11 position at the U.S. Army Training and Doctrine Command in Fort Knox, KY [40].

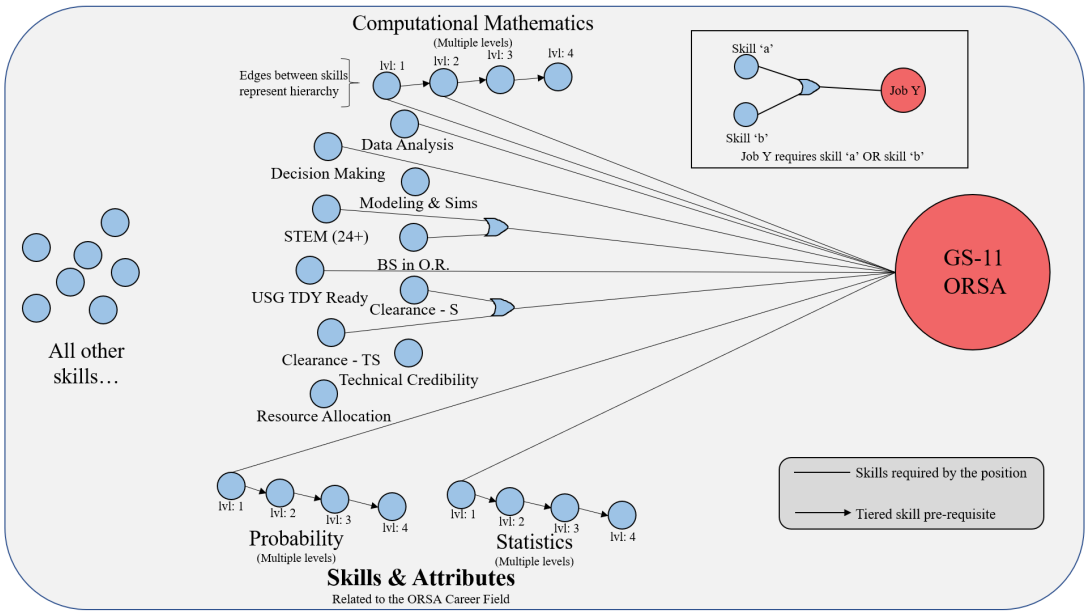


Figure 3.4. Example Subgraph of the Skills Related to a Specific Position

In this example subgraph, we see a specific position from the Operations Research Systems Analyst (ORSA) career field related to 11 different skills. The skills associated with this position come directly from the job requirements of the position as written in the job

posting or job description. Some required skills are of the singular attribute type mentioned above, such as "Temporary Duty (TDY) ready", while other skills have been related to a sequence of skills. The job requirement 'having a strong background in mathematics' has been subjectively related to a level 2 proficiency in computational mathematics. Some positions require attributes that can be categorized as ordinal data and eclipsed by a greater attribute. In this example the position requires a "Secret" security clearance. However, a "Top Secret" clearance will supersede the requirement for a "Secret" clearance. Finally, some job skill requirements can be substituted. This position's job requirements offers job seekers the choice of either a bachelor's degree in Operations Research, or 24 or more credits in Science Technology Engineering and Math (STEM) courses.

3.3.3 Mapping Content to Skills

Further expanding on the definitions introduced in Section 3.3.1, this section defines the properties of the training content vertices and the relationships between training content and skills. The collection of training content depicted in the CTN is organized hierarchically by subject matter, mirroring the existing schema of the CHUNK Learning Network described in Section 2.3. The collection of training content in this framework, illustrated here in Figure 3.5, is oriented towards skill and talent development. As such, we adjust the definitions from Table 2.1 to reflect professional development nature of the training content to include more than just academics.

The hierarchical collection of skill development content is organized from the top down. Overarching subject areas are defined as "units" and are represented by a *unit* vertex. These could be similar to what is covered in a long-term training course that covers multiple training topics. Since one of our research goals is to only provide relevant training, and units are broad in nature, units are only used for content classification and organization and would normally not be included in any individualized training plans. Each unit module contains a set of chunk modules, represented by *chunk* vertices. These chunk modules are major topics associated with the unit's subject area. Similarly, each chunk module contains a set of chunklet modules, represented by *chunklet* vertices. chunklets are smaller sub-topics that are be part of the parent chunk topic (or lessons in an academic setting). Finally, each chunklet module contains a set of activity modules represented by *activity* vertices. Each activity vertex represents one training artifact from a wide variety of training materials such

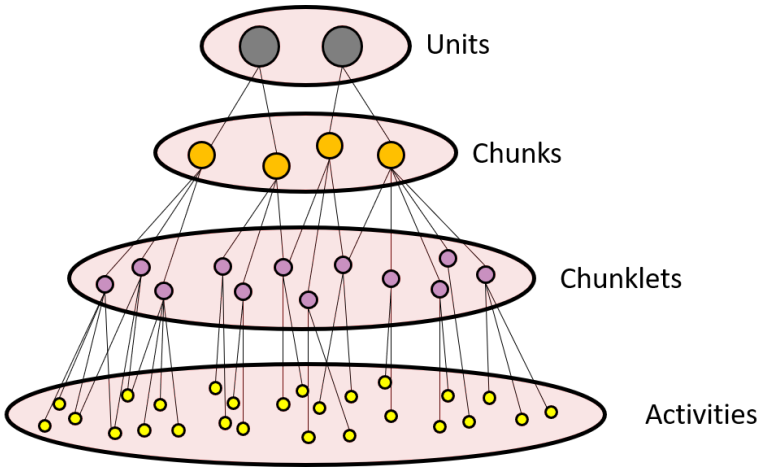


Figure 3.5. Training Content Hierarchy: units contain chunks, which contain chunklets, which contain activities

as: a web page, a chapter of text, tutorial video clip, lecture segment, code, or interactive exercise.

In addition to the hierarchical relationships described above, it is possible for sub-topic or chunklets or activity modules to be reused in multiple topics such as having several chunklets on the same idea, but each presented either for beginner, intermediate or advance learner. Therefore, a chunklet (or activity) vertex could be edge-related to multiple chunks (or chunklets). Content can also be related by pre-requisite content, similar to courses in an academic setting requiring another course before being eligible for enrollment.

Activity Vertex Attributes

As the base layer of the training hierarchy, the *activity* vertices host all training material and content metadata referenced by the recommender system later in Section 3.5. Similar to a user's κ KLS vector, each activity node is assigned a $1x4$ vector (annotated as δ ("delta")) which describes the activity's ELM mode of presenting training material. Content creators and authors can approximate the activity's δ value each time a new activity module is added to the content library based on the modules ELM training mode. This value can then be compared to the learner's kappa values, for the purpose of identifying the most relevant learning content for each user.

$$\delta = [\text{CE, RO, AC, AE}]$$
$$CE + RO + AC + AE = 1$$
$$\text{CE, RO, AC, AE} \in (0, 1]$$

Using the mode descriptions from Section 2.2.2 as a guide, each activity is described by the portion of its content that matches each training mode. If an activity is mostly RO with little to no experimentation, then the second entry of the δ vector will be much larger than the fourth; if the same module uses a mixture of conceptualization and CE, then the first and third δ entries will be similar and the activity's actual δ vector values may be close to $[.25, .5, .25, 0]$. Ultimately, the content authors and system administrators will have to subjectively determine the δ values for each activity module as our model currently does not propose a method for objectively determining the ELM modes.

In addition to the δ vector, each activity has a computed average quality rating, a unique completion credit value for each chunklet that the activity is related to, and Boolean flags to indicate if the activity has already been completed or if it is being included in the recommender system's set of proposed training content. These attributes are further described in Section 3.5 and Table 3.2.

Skill Mapping

One critical component that this framework requires is a sufficient set of training content to train each of the skills in the network. We make the assumption that the system incorporates a sufficient number of training modules such that a surjective mapping of training topics (chunks) onto skills exists. Furthermore, this assumption also implies that there also exists enough related activities to sufficiently train each chunklet, and enough related chunklets to sufficiently train each chunk. When this assumption is satisfied, a surjective mapping, similar to the one shown in Figure 3.6 exists between every skill node and a set of chunk nodes.

The chunk level of the content hierarchy was chosen for the mapping between skills and content as units often cover more material than necessary for a specific skill.

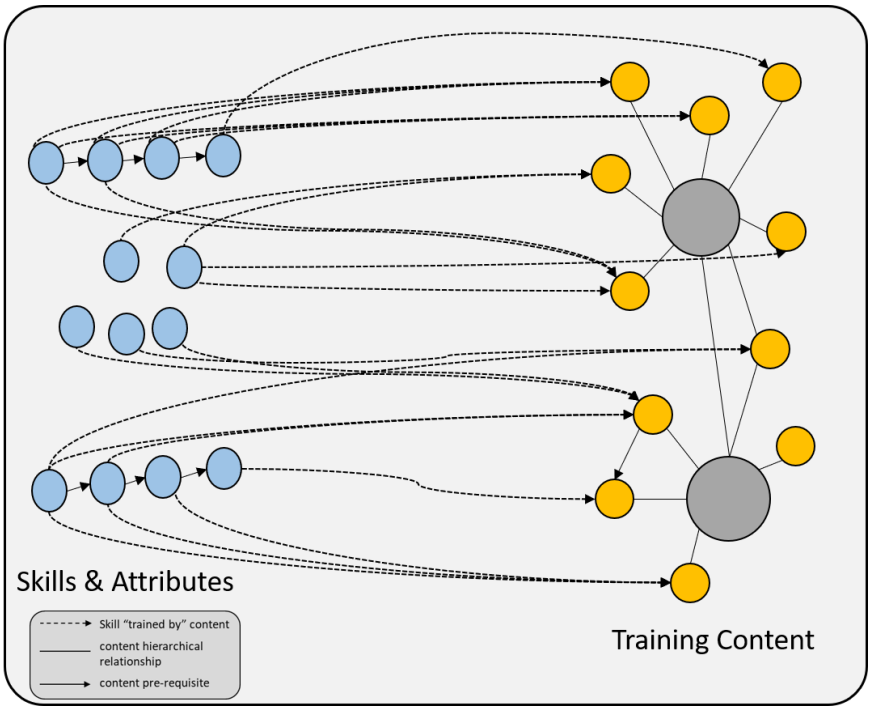


Figure 3.6. Subgraph Demonstrating Surjective Mapping of Content onto Skills

As seen in the example in Figure 3.7 a single skill ("Computational Mathematics: level 3") is directly related to 12 different chunks of relevant training content. Five of the chunks are required training content for the skill. A job seeker would have a pair of "either/or chunk" options, and one "pick two of three chunks" option. Furthermore, required chunks have pre-requisite chunks which would also have to be completed, even though they are not directly related to the skill.

The systematic creation, organization, and management of the training content library is a vast challenge and is outside the scope of this thesis. This thesis assumes that such a library of content can be established and maintained using the CHUNK Learning Network model as a proof of concept. While the recommender system does not incorporate deep content metadata such as keywords or subject areas, it is expected that such data is instrumental in the implementation of the training content topography. Furthermore, this research assumes that each content node is built and maintained by its respective author, and that content authors are able to assign required subordinate content relationships appropriately.

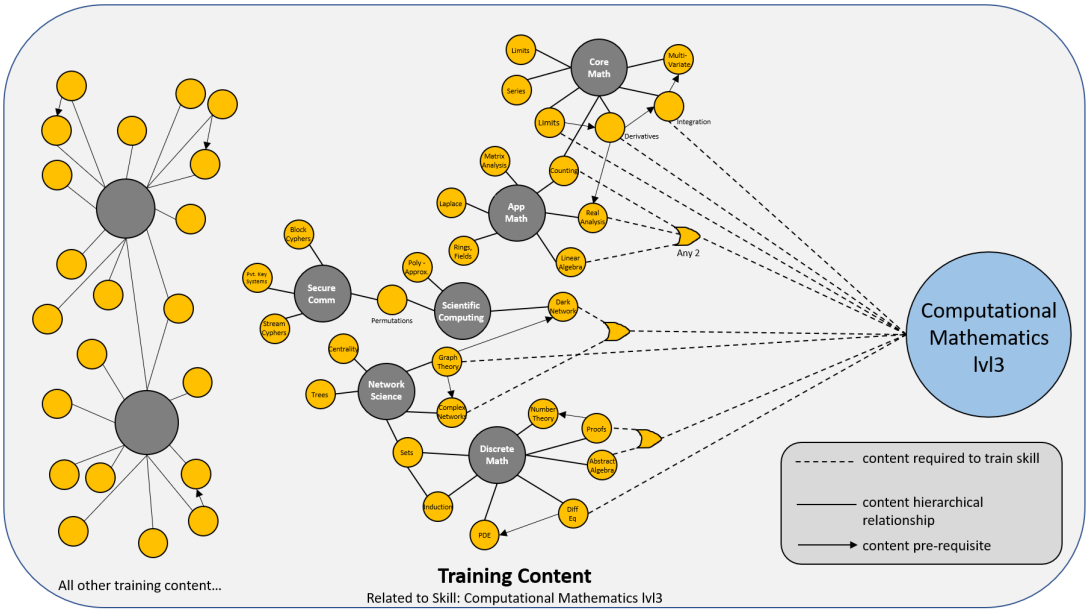


Figure 3.7. Example Subgraph of Training Content Related to a Skill

3.4 Step 2: Graph-Based Training Database

In order to effectively implement a recommender system and enable interactive user actions, we form the CTN as a graph-based database. The graph-based database is essential to the recommender system described in Section 3.5. A graph-based database facilitates efficient storage and recall of information on relationships, rather than on CTN entities alone. A single graph-database represents all the elements and metadata from the CTN. This graph also serves as a basis for building and displaying unique user training networks. Some training content (texts, exams, documents, lectures, etc.) may be embedded with the associated activity objects on the graph-database DBMS. Other training content may only reference from the activity vertices (such as websites, large video files, etc.).

3.4.1 Forming the Database with Neo4j

For our work, we form the CTN as a graph-based database using the Neo4j Graph Data platform [41]. The proposed model is mathematical in nature and can be implemented with any preferred compatible software suite. However, because software suites manage data and transactions differently, we want to explicitly describe the tools we are using and

the reasons why we believe they support our methodology. Thus, we develop our model using the following Neo4j tools based on their availability and effectiveness in running the proposed model's algorithms.

Neo4j DBMS

The Neo4j graph-based database platform, introduced in Section 2.1.3, is a commercially developed software suite designed for individual users and institutions alike. The Neo4j platform is available at no cost to individual users. This allows us to develop and test models using a local computer as the DBMS host server. Neo4j is also a "cloud-friendly" platform. Cloud-based applications allow remote access to the model for multiple users without requiring users access to local systems.

Cypher Query Language

Neo4j uses a unique graph-database query language called Cypher Query Language (CQL). CQL is an open-source query language unique to Neo4j inspired by SQL. CQL is more efficient than SQL for graph databases because it uses simplified ASCII symbol pattern recognition to search for graph patterns rather than joining tables and evaluating data for records that meet desired criteria [42]. The example CQL queries in this chapter are written with a specific structure. MATCH: the pattern we are looking for in the graph, this could include specific or general nodes annotated with parenthesis, or specific or general relationships annotated with squared brackets. Then we can either RETURN data associated with the graph entities, or we can modify the graph through CREATE or UPDATE type commands. Most database transactions are accomplished through a single line CQL query.

Bloom Viewer

The Neo4j platform includes a visual graph exploration application called Neo4j Bloom. This application allows us display training networks and their attributes while interfacing directly with the Neo4j DBMS. Since this application is integrated with the Neo4j software suite, this tool allows us to develop the model visually without the need to export data.

Neomodel Python Translator

We incorporate python scripting to systematically build the Neo4j DBMS through the Neomodel python package. We also use the python scripting to import data into the DBMS once it is running. The Neomodel python package facilitates the interface between a python environment and a Neo4j DBMS through translated CQL queries. This enables python scripting to build synthetic database graphs and systematically import new data from external sources. Since Neomodel is a bi-directional DBMS interface, it also facilitates third-party graph analysis.

3.4.2 Primary Database Nodes

A Neo4j DBMS graph consists of "nodes" and "relationships" between those nodes. Just as the CTN can be expanded to include additional modifying elements, the core database graph can be expanded to include additional modifying elements as well. In this section we define the seven primary node types used in our proposed model (shown in Figure 3.8). We also introduce the primary attributes associated with each node required by the model. Additional attribute details are shown in Table 3.2 (in Section 3.4.3). For added clarity in this work, when referencing a specific database graph object such as a *user* node we will label it with italicized text, and when referencing the generic noun or the represented network entity (i.e., system users) we will use un-italicized text.

Node: System User

Graph *user* nodes represent the job seeker users from the training network and have two primary attributes: κ and ψ ("psi"). We represent the user's learning style preferences as κ (described in Section 3.3.2). We dynamically update the user's learning preferences through the modifier, ψ, as the user completes training and provides feedback, (ψ us described later in Section 3.5.3)

Node: Position

Graph *position* nodes represent the specific jobs or positions from the training network. *Position* nodes have no unique attributes other than identifying information and their relationships.

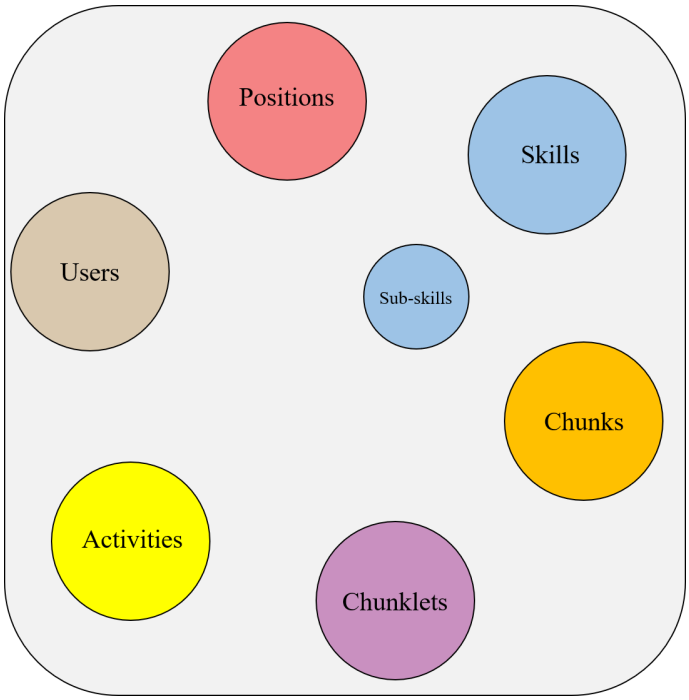


Figure 3.8. Database Graph Nodes

Node: Skill and Sub-Skill

Graph *skill* nodes represent the skills and attributes required by positions from the training network. *Skill* nodes have one primary attribute: ϕ ("phi"). ϕ measures the average quality rating of the content modules which train this skill. *sub-skills* are built identically to *skill* nodes, but act as an intermediary level between *skills* and *chunks* in some cases (described later in Section 3.5.4).

Node: Activity

Graph *Activity* nodes represent the individual professional skill development artifacts from the training network. These base layer content hierarchy nodes either hold training content, or point to external content that user can complete. *Activity* nodes have five primary attributes: ϕ, ω ("omega"), υ ("upsilon"), ρ ("rho"), and δ:

- ϕ: The average quality rating of the activity module.
- ω: A computed "match score" between this activity and the user. A detailed description of how this score is computed is covered in Section 3.5.4.

- υ: A Boolean flag indicating if the user has completed the content module (see Section 3.5.2 for implementation in the model).
- ρ: A match score bonus applied when the *activity* supports the training in multiple relevant *chunklets*
- δ: The content module's ELM learning mode vector (introduced in Section 3.3.3).

Node: Chunklet

Graph *chunklet* nodes represent the chunklet vertices from the training network. *chunklet* nodes have five primary attributes:ϕ, ω, υ, ρ and ν ("nu").

- ϕ: A chunklet's ϕ value is the average of the related activity nodes' ϕ values.
- ω: chunklet "match scores" are computed similarly to the activity 'match scores'.
- υ: A Boolean flag indicating if the user has completed the content module.
- ρ: A match score bonus applied when the *chunklet* supports the training in multiple relevant *chunks*
- ν: A composite ELM learning mode vector (described in Section 3.5.3).

Node: Chunk

Graph *chunk* nodes represent the chunk vertices from the training network. *Chunk* nodes are also the highest echelon of the training content hierarchy represented in the GOOD graph. *chunk* nodes have five attributes: ϕ, ω, υ, ρ, and ν.

- ϕ: A chunk's ϕ value is the average of the related *chunklet* nodes' ϕ values.
- ω: chunk "match scores" are computed similarly to the activity "match scores".
- υ: A Boolean flag indicating if the user has completed the content module.
- ρ: A match score bonus applied when the *chunk* supports the training in multiple relevant *skills*
- ν: A composite ELM learning mode vector.

Together these seven node types represent all the significant influences from the personal knowledge and skill acquisition environment. The data contained in the attributes quantifies all the necessary information about each of the nodes in a relatively small data set. This organization of information into structured nodes within the graph-database will allow the recommender system in Section 3.5 to focus the library of professional development content for job-seekers.

3.4.3 Primary Database Relationships

In this section we define the eight relationship (edge) types used in the database graph (shown in Figure 3.9). In the previous section, we defined different node types to manage what data would be stored where, and provide simplicity for data recall. In this section we define relationship types for the same reason. Pattern matching in graph-databases is designed to be intuitive, by labeling the relationship types we can explicitly decide their behavior and associated data.

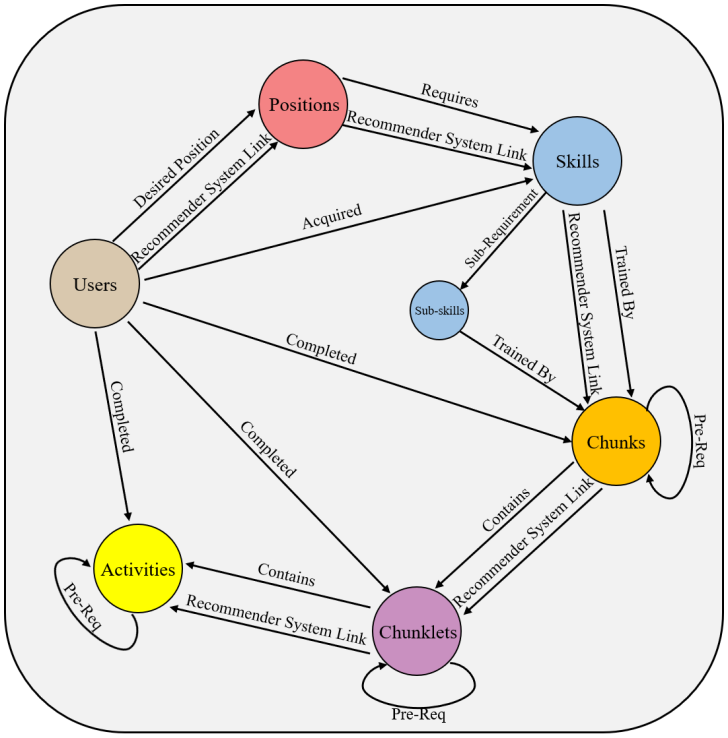


Figure 3.9. Graph Database Schema in Neo4j DBMS

In this section we also introduce the primary attributes associated with each relationship required by the model. Additional attribute details are shown in Table 3.2 (in Section 3.5.2 where they are computed.) We explicitly define these eight relationship types since each structured relationship type in the database is associated with a different set of attributes. Defining different relationship types in the database also supports quicker pattern matching when conducing data recall and transactions.

Relationship: DESIRED_POSITION

Directional *DESIRED_POSITION* relationships exist between user nodes and their desired position(s). These relationships have no unique attributes other than identifying information.

Relationship: REQUIRES

Directional *REQUIRES* relationships exist between position nodes and any skill nodes required by those positions. These relationships have no unique attributes other than identifying information.

Relationships: TRAINED_BY & CONTAINS

TRAINED_BY & CONTAINS relationships form the structure of the training content hierarchy. Directional *TRAINED_BY* relationships exist between skill or sub-skill nodes and any chunk nodes required by the skill. A directional *TRAINED_BY* relationship also exists between skill and sub-skill nodes when there is a choice of chunks required by a skill. *CONTAINS* relationships exist between parent chunk or chunklet nodes and child chunklet or activity nodes respectively. Together these two relationship types identify the hierarchical relationships of the training content. Both *TRAINED_BY & CONTAIN* relationships have three primary attributes: α ("alpha"), β ("beta"), and γ ("gamma").

- α: A Boolean flag to indicate if the child node has been selected by the recommender system for inclusion in the parent node's content mix.
- β: A Boolean flag which indicates that the recommender system must include the child node in the parent node's content mix regardless of learning style preference.
- γ: Parent node completion percentage. Each child node has an independent γ value for each parent node. In order to complete a parent node, a user must have completed enough child nodes such that the summation of their γ values is ≥ 1

Relationship: PRE_REQ

Directional *PRE_REQ* relationships identify a content pre-requisite requirement between two content nodes (generally of the same hierarchy level). These relationships have no unique attributes other than identifying information.

Relationships: ACQUIRED & COMPLETED

Directional *ACQUIRED* relationships exist between user and skill nodes. Once a user has completed a sufficient set of chunks required by a skill node, an "Acquired" edge is established between them. Directional *COMPLETED* relationships exist between user nodes and content nodes. Once a user has completed a sufficient number of contained training modules, a *COMPLETED* relationship is created. These relationships have no unique attributes other than identifying information.

Relationship: REC_SYS_LINK

REC_SYS_LINK relationships are created as part of the output of the recommender system in Section 3.5.4. They form the edges of the paths between the users and every node the recommender system has deemed as a recommended training content module. These relationships have no unique attributes other than identifying information.

Table 3.1. Relationships Defined in the Model Network (depicted as edges)

Database Graph Relationship Types			
Label	From	To	Relationship Description
COMPLETED	User	Content	Relationship between users and content sufficiently completed.
DESIRED_ POSITION	User	Position	Directed edge(s) which identify the user-position combinations used by the Recommender System
ACQUIRED	User	Skill	Directed edges identifying the skills each user inherently possesses or has acquired.
REQUIRES	Position	Skill	Directed edges between each position and all the skills required by the job description.
TRAINED_BY	Skill	Content	Mapping between each skill and the CHUNK vertices which train the skill.
CONTAINS	Content	Content	Directed edges identifying the parent-child relationships within the content hierarchy.
PRE_REQ	Content	Content	A content module may require the completion of another as a pre-requisite at the same (or sometimes lower) hierarchy level.
REC_SYS_LINK	Any	Any	Recommender System created relationships between a user, their desired position, skills and training content.

3.5 Step 3: Training Content Recommender System

The CHUNK-PD framework implements a hybrid recommender system to produce individualized training plans for each user. Our hybrid recommender system combines the approaches of collaborative-filtering and content-based recommender systems (see Section 2.4 for definitions). We combine these two approaches to account for both user preferences and content features, while employing aggregation functions for efficient computation. The recommender system executes four core functions in sequence to identify an optimal set of training content for each user to complete. The recommender system requires an identified "user and desired position" combination and an active connection to the training network database graph; it then employs its four core functions as shown in Figure 3.10.

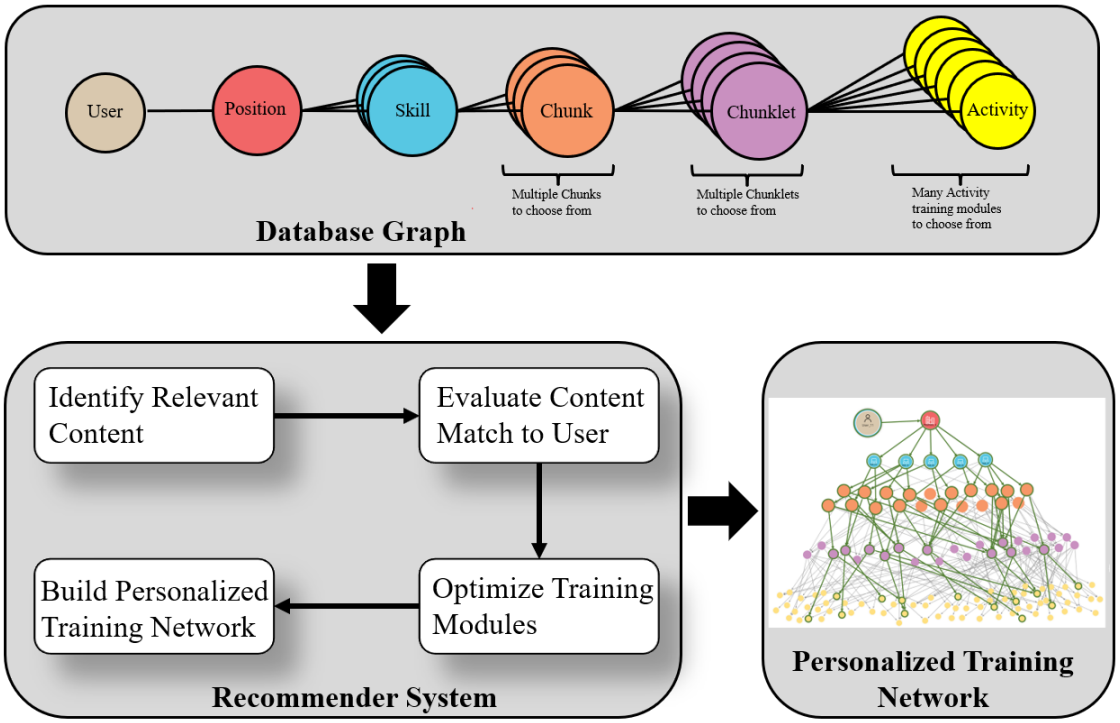


Figure 3.10. Recommender System and its Four Functions

As its core purpose, the recommender system picks the best combination of activity modules for each chunklet, chunk, and skill based on the user's learning style preferences and the quality of the content. Our methodology builds on each users' experience in the system. That is, as the user completes some training modules, the recommender system incorporates

the user's feedback to refine results moving forward. Additionally, the recommender system incorporates the users' assessments results and potential future desired positions to refine its recommendations.

In contrast to the traditional style of instructionism where curricula are designed from the top down and the only feedback required is through examination after the material is taught. This system combines top-down skill training identification with bottom-up refinement prior to training in order to provide a set of training paths that are both meaningful and engaging for each job-seeker.

First, the recommender system conducts a top-down training path exploration through the content hierarchy to identify a subgraph of relevant training content. Next, the recommender system applies the user's attributes, acquired skills, and any previously completed modules to the relevant content subgraph. Third, the recommender system conducts a bottom-up evaluation of training content. Finally, the recommender system produces a unique personalized training network.

3.5.1 Function 1: Top-Down Breadth-First-Search

The recommender system's first function reduces the database down to a subgraph of training content relevant to the position(s) desired by the job seeker. Rather than conduct analysis over then breadth of the database graph, we establish subsets of relevant training content for each user using a modified BFS algorithm. The top-down BFS identifies paths through the training content from the current *user* to every reachable end point (the *activity* nodes relevant to the desired position(s)). The BFS algorithm returns four sets of relevant content: *skills*, *chunks*, *chunklets*, and *activities*. The function output is creating or updating a subgraph in the database based on the relevant content for the user-position combination.

The sub-steps of Function 1 set the conditions for recommender system computations by identifying a user and desired position that forms the input for the pattern matching algorithm that returns a subgraph of relevant training content. This is done through the following three sub-steps:

1. Create or update a *DESIRED_POSITION* relationship between the *User* and *Position* nodes.
2. Implement a modified BFS to identify relevant training modules.

3. Update or create a relevant content subgraph based on the connected components identified by the BFS.

We now introduce the details on each of the sub-steps.

Create *DESIRED_POSITION* Relationships

Up until this point, the model is generalized in nature. We define the CTN to represent the environment and we establish a graph database to capture CTN elements and store data. However, once a job-seeking user identifies one or more desired jobs the recommender system begins computing and storing graph data specific to the user and their desired position. By reducing the scope of computation to only the limited subgraph of relevant data, the recommender system can be more efficient in returning training plans for each user.

Desired Positions are identified as inputs from the user. Then each user can select from any existing position in the CTN. Users typically pick only one desired position, but they can pick more than one to see training available options for a series of positions in the future. A single database transaction removes any existing *DESIRED_POSITION* relationships, and creates a *DESIRED_POSITION* relationship between the *user* node and the indicated *position* node(s).

Implement the Modified Breadth-First-Search Algorithm

The second sub-step of Function 1 executes a modified BFS algorithm to build sets of required skills and relevant training content. Our BFS algorithm uses the *user* as the start point and identifies any skills reachable through the desired position(s). The algorithm then searches for any reachable *chunks* from the identified *skills*. This repeats for any reachable *chunklets* and *activities*. The difference in our BFS from the one in Section 2.1.2 is our algorithm records all paths to reachable nodes, not just the first (shortest) path to reachable nodes. The output of the BFS algorithm, as seen in Figure 3.11, is a subgraph of training content relevant to the skills required by the position(s) desired by the job-seeker.

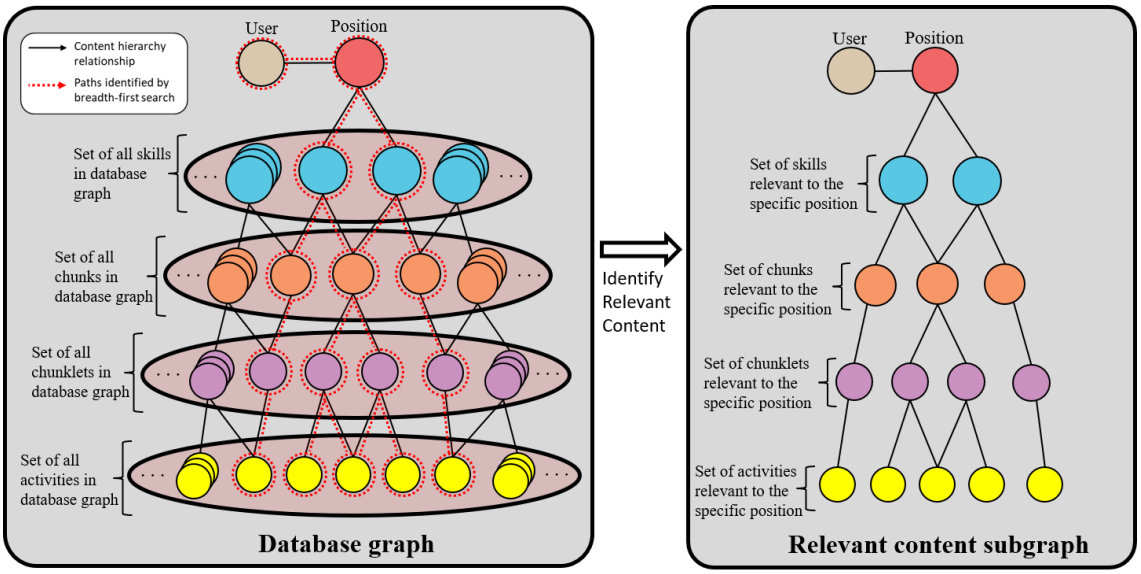


Figure 3.11. Function 1: Breadth-First-Search

The BFS is executed within the database environment through a series of CQL queries. Each query returns a set of *skills* or content nodes, which relate to the required skills for the desired position(s) or relevant training content modules. The following CQL query will return a set of required skills related to the specific user node (identified by user_id):

```
MATCH (user{uid:"user_id"})-[:DESIRED_POSITION]-(:position)-[:REQUIRES
    ]->(n:skill) RETURN n
```

Create a Relevant Content Subgraph in the DBMS

The final sub-step in Function 1 is creating a subgraph of relevant content in the DBMS for the model user. This subgraph stores the model data associated with each user's skill identification and training process.

3.5.2 Function 2: Evaluate Training Content

The recommender system's second function incorporates the unique attributes and accomplishments of the user into the evaluation of relevant training content identified in Section 3.5.1. Function 2 creates or updates recommender system variables listed in Table 3.2 prior to content optimization in Function 3. This function's goal is to produce an

accurate prediction of how well each activity module compares with the user's preferred learning style.

In order to make the aforementioned prediction, this function first needs to update the parameters which feed into the match score computation. This function first interacts with the subgraph of relevant content to read and update the local variables listed in Table 3.2 then computes updated values for the supporting variables needed to compute the match value ω for each activity. The sub-steps of Function 2 are as follows:

1. Update the user's subgraph based on acquired skills and completed training content.
2. Update content quality ratings.
3. Update *user's* learning style refinement vector, ψ, based on completed training modules.
4. Update content multi-use bonus modifier ρ.
5. Create or update the ω match score for every *activity* in the subgraph of relevant content.

Table 3.2. Variables Used in Recommender System Optimization

Variable	Type	Applied to	Description
α	Boolean	Relationships between content	Variable set by recommender system for each user-position evaluation. Set to true if the child content is recommended for inclusion for the parent node.
β	Boolean	Relationships between content	Required content flag. True value indicates that the child node is required by the parent content node
γ	Float	Relationships between content	Completion credit: indicates the relative percentage of the parent node's training thresh hold that the child node trains.
δ	$1x4$ vector $\in [0, 1]$	Activity vertices	Four dimension position vector describing the training activity's ELM mode of learning. Scaled such that the sum of δ elements = 1
κ	$1x4$ vector	User vertices	Four dimension position vector describing the user's learning style preference
ψ	$1x4$ vector $\in [-1, 1]$	User vertices	Four dimension learning style preference refinement
ν	$1x4$ vector $\in [0, 1]$	*chunklet* & *chunk* vertices	Computed four dimension position vector describing content's weighted average ELM mode of learning based on selected subordinate content ELM values
ω	Float	Content vertices	User-Content matching score.
ϕ	Float $\in [1, 5]$	*Content* and *skills*	Average quality rating for content (on a 5-star scale)
ρ	Float	Content vertices	module multi-use bonus
ε	Float $\in (0, .25)$	*Activities*	'tunable' multi-use bonus parameter. As ε increases, the recommender system puts more emphasis on selecting content which is seen in multiple modules.
υ	Boolean	Content Vertices	Logical flag indicating if the current *user* has completed the training module (1=incomplete)

Update Acquired Skills and Completed Content: υ

As the user heuristically views and finishes training modules, the recommender system must account for completed training modules that are relevant to the identified skills for that user. Prior to computing (or updating) the user's personalize training network, recommender system Function 2 checks for completed training content modules and newly acquired skills. Function 2 creates or updates *ACQUIRED* relationships as necessary in the user's subgraph.

The recommender system accounts for completed training content each time the system updates a user's personalized training network. Removing completed modules from consideration would skew the optimization outcomes. Rather, we want completed modules to provide credit to any parent module in the content hierarchy. To account for completed training in recommender system computations, Function 2 assigns the completed modules a cost of zero in the optimization process (in recommender system Function 3.) The Boolean flag υ, assigned to any content node, indicates completeness status of training (0=completed, 1 = incomplete). The υ value is stored directly on content nodes (not the relationship edges) and is updated as part of recommender system Function 2 prior to the optimizations in the next function.

Content Quality Ratings: ϕ

As a user completes training modules, they have the opportunity to provide feedback in the form of a quality rating. Users are prompted to provide a one-to-five star rating for training activities as they complete them. The average rating for each activity is computed and stored as *activity* node attribute ϕ. All *activity* nodes start with a default rating of $\phi = 3.0$ to prevent cold start computational errors. These ratings are updated each time a user completes the activity.

Adjust User Learning Style Preference: ψ

As users complete training content modules and provide feedback in the form of quality ratings, the recommender system adjusts the user's perceived learning style to better match content. This learning style adjustment factor is annotated at ψ, and like the *user's* κ attribute, ψ is a $1x4$ vector. Unlike κ, ψ can contains positive or negative values to refine κ accordingly. For *user* node u, learning style refinement ψ is computed as follows:

$$\psi = [1, 1, 1, 1] + \frac{\sum_{i \in N} \Delta\phi\delta_i}{4 \cdot \text{length}(N)} \tag{3.1}$$

where:

ψ = user learning style refinement vector

N = the set of *activities* i completed and rated by user u

$\Delta\phi = \phi_u - \phi_i$ (the difference between the user's rating and the average rating for i)

ϕ_u = user u's rating for activity i

δ_i = *activity* i's ELM training mode

To prevent excessive refinement from a small N, each *user*'s $\psi = [1, 1, 1, 1]$ until $N \geq 10$.

Content Multi-Use Bonus: ρ

The recommender system has an additional feature to favor training content which supports multiple parent training modules. What this means, is that when given the option between two otherwise equivalent activities for a chunklet, the optimization function will recommend the activity that is contained in $i + 1$ or more relevant chunklets over the activity that is only contained in i relevant content, for $i \in \mathbb{N}$. Lastly, if there is even a tie in this case, then the system randomly picks one activity. These multi-use match value bonus scalars (annotated as ρ) are tune-able parameters which can be adjusted for greater or less sensitivity as the user desires. A larger bonus value ρ makes the content module a better match to any user.

$$\rho = (1 - \varepsilon)^{\frac{l}{len(L)}} \tag{3.2}$$

where:

ρ = multi-use match value bonus scalars

ε = maximum bonus size (tunable parameter, default value = 5%) $\in (0, 0.5]$

l = the number of related parent nodes in the set of relevant content

L = the set of relevant training content nodes one tier up

Content Match Score: ω

The content match score (ω) is a composite value that helps the recommender system predict how well the content module will match the user's profile. This match score is a combination of the users learning style preference, the content delivery mode, and the perceived quality of the training content module as we show next. Seen in Equation 3.3, we compute ω by determining the Euclidean distance between the module's ELM training mode δ, and the user's ψ modified κ learning style preference. Then we scale this distance by the inverse of the perceived quality value ψ and apply any multi-use bonus ρ. Subjectively, a lower ω score indicates a better match and is preferred in recommender system computations. For *user u* and *activity i*, ω is computed as follows:

$$\omega_i = \frac{\rho_i}{\phi_i} \|\delta_i - \psi_u \odot \kappa_u\|_2 \tag{3.3}$$

where:

ω_i = content match score for module i

ρ_i = multi-use match value bonus for module i

ϕ_i = quality rating for module i

δ_i = ELM content delivery mode for module i

ψ_u = learning style refinement vector for user u

κ_u = learning style for user u

These updated ω values are stored directly on the *activity* nodes in database graph.

3.5.3 Function 3: Bottom-Up Content-User Matching Evaluation

This is the core function of the recommender system algorithm. Function 3 performs a bottom-up optimization of training content module composition for *chunklets*, *chunks*, and *skills* shown in Figure 3.12. Function 3 determines the combination of child nodes for each parent chunklet, chunk, and skill node based on the match scores through a series of Linear Optimization Programs (LPs). The output of this function is an optimized set of content which will complete the user's training on skills required for the user's desired position.

The sub-steps of Function 3 are as follows:

1. Optimize the set of recommended activity modules for every relevant *chunklet* node.
2. Update composite variables ν, ω, and ϕ for each *chunklet* based on recommended activities.
3. Optimize the set of recommended chunklets for every related *chunk* node (based on the activities recommended in sub-step 1.)
4. Update composite variables ν, ω, and ϕ for each *chunk* based on recommended chunklets.
5. Optimize the set of recommended chunks for every required skill not yet acquired (based on the chunklets in sub-step 3.)

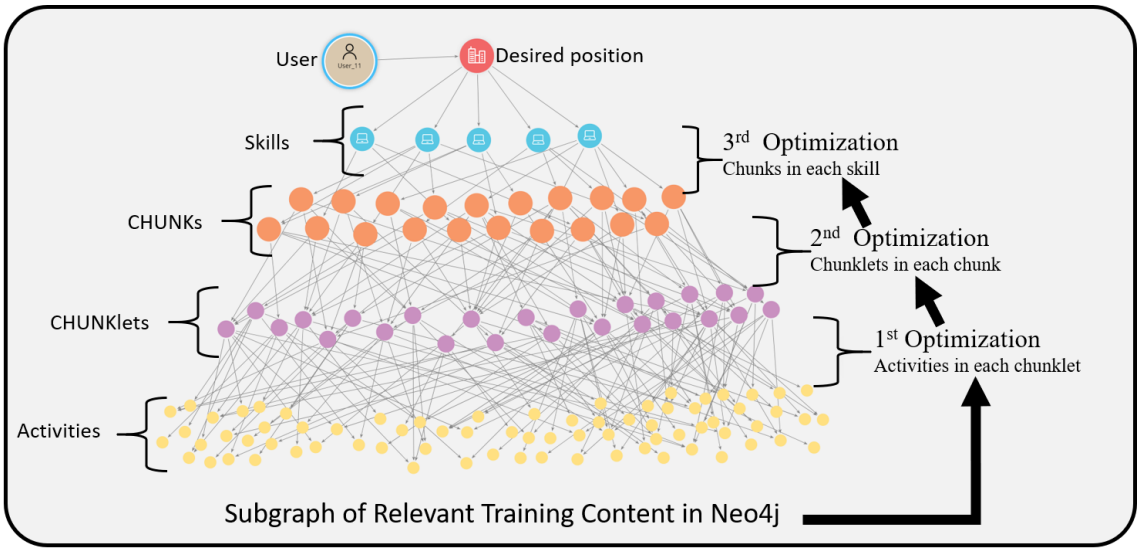


Figure 3.12. Function 3: Bottom-Up Optimization: identify activities to be recommended to the user, then identify the chunklets that contain the recommended activities, then the chunks based on the recommended chunklets and skills needed to be acquired

To provide clarity, we redefine the term *recommended* to emphasize that every parent node is adjacent to a combination of optional and recommended child-nodes. Furthermore, since the system allows job-seekers to heuristically explore content, this labeling of recommended content assists the job-seeker to identify the best selection of skill development training at each tier of the hierarchy.

Definition 3.5.1 *recommended content*

> *Sub-modules that the recommender system has explicitly selected for a job-seeker to complete in order to satisfy the completion requirements of a parent skill, chunk, or chunklet module.*

Each parent skill, chunk and chunklet is comprised of optional and recommended sub-modules. The optimal set of sub-modules to complete the parent module is labeled as recommended content and the remaining sub-modules are labeled as optional.

This recommender system function is the computational heart of our proposed model. This function computes the data stored in all of the recommender system variables (listed again in Table 3.2) to determine an optimal combination of sufficient training content for the required skills based on user learning style and content quality.

Linear Optimization of chunklets

Function 3 re-iteratively calls Linear Optimization Program 1 (LP1) for each *chunklet* in the subgraph of relevant content. LP1 evaluates the set of related *activity* nodes that have a *CONTAINS* relationship to each *chunklet* in order to determine which activities should be recommended content modules. LP1 then updates the α Boolean markers on the associated *CONTAINS* relationships, to indicate which activities comprise the optimal content mix for the parent chunklet training module. LP1's pseudo code is as follows:

Linear Program 1 **(NPS Standard Format [43]):**

Sets and Indices:

$n \in N$ Set of related *activity* nodes

Data:

β_n	β value of n; (1=required by *chunklet*)	$\beta \in \{0, 1\}$
ω_n	ω value of *activity* node n	$\omega \in \mathbb{R}^*$
υ_n	υ value of *activity* node n	$\upsilon \in \{0, 1\}$
γ_n	γ value of relationship n	$\gamma \in \mathbb{R}^*$

Decision Variables:

α_n Which *activity* nodes to recommend (indicated by $\alpha = 1$)

Objective Function:

$$\min_{\alpha} \sum_{n \in N} \alpha_n \omega_n \upsilon_n$$

Constraints:

$s.t.$

$$\sum_{n \in N} \alpha_n \gamma_n \geq 1 \qquad \text{(total completion credit of recommended activities must be } \geq 1)$$

$$\alpha_n - \beta_n \geq 0 \quad \forall n \in N \qquad \text{(if activity } n \text{ is required, } \alpha_n \text{ must = 1)}$$

$$\gamma, \omega \geq 0$$

$$\alpha, \beta, \upsilon \in \{0, 1\}$$

LP1 can be implemented to determine an optimal solution using any preferred method of linear optimization such as the Simplex method [44]. We suggest using an automated solver such as "CBC" with the python package "Pyomo". CBC is an open-source mixed integer linear programming solver [45] and Pyomo is an object-oriented algebraic modeling package [46], [47]. Both CBC and pyomo are readily available online at no cost.

Once the solver has converged to an optimal solution, the model's values for α correlate to which activity modules are considered recommended for the given *chunklet*. In order to save these α values, we assign them back onto their respective *CONTAINS* relationships (between the *chunklet* and *activity* nodes) with an iterated database transaction such as:

```
for r in R  # R is the set of relationships in LP1:
    MATCH ()-[relationship:CONTAINS]-() WHERE id(relationship)=
   r_IDnumber SET relationship.alpha = newAlphaValue
```

In this manner, we can refer back to which activities are recommended for each specific chunklet within the subgraph without needing to hold the information in memory or pass the complete set of data forward with each computation.

Composite Attribute Updates

After completing the round of chunklet optimizations using LP1 (or chunks with LP2), the recommender system needs to update each parent *chunklet* (or *chunk*) nodes' attributes before moving on to the next tier of optimizations. After LP1, each chunklet only requires a subset of related activities to be completed in order to complete the chunklet. Therefore, considering only the recommended *activities* for each *chunklet* we can create or update each *chunklet's* ν, ω, and ψ attributes.

The composite ELM-based training mode (ν) for each *chunklet* node is computed as a weighted average of each recommended *activity* node's ELM training mode. Each *activity's* δ is weighted by its contribution to the *chunklet* (zero if α=0, and γ otherwise).

Computing ν for *chunklets*:

$$\nu = \frac{\sum \alpha_i \gamma_i \delta_i}{\sum \alpha_i \gamma_i}, \forall i \in \{\text{set of } \textit{activities} \text{ related to the } \textit{chunklet}\} \tag{3.4}$$

Computing ν for *chunks* is performed similarly, but with the substitution of ν for δ:

$$\nu = \frac{\sum \alpha_i \gamma_i \nu_i}{\sum \alpha_i \gamma_i}, \forall i \in \text{set of } \{\textit{chunklets} \text{ related to the } \textit{chunk}\} \tag{3.5}$$

Next, we update each *chunklet* (or *chunk*) ϕ quality rating. For *chunklets* and *chunks*, ϕ is computed as a summed weighed average of the quality ratings of their child content nodes. The new ϕ values are updated and stored directly on the *chunklet* or *chunk* nodes where they will be used to update the match scores. The composite ϕ is computed for *chunklets* and *chunks* in the same manner:

$$\phi_{\text{parent node}} = \frac{\sum \alpha_i \gamma_i \phi_i}{\sum \alpha_i \gamma_i}, \forall i \in \{\text{set of child nodes related to the parent node}\} \tag{3.6}$$

Finally, we can compute new ω user matching scores for the optimized parent content nodes (*chunklets* or *chunks*). *chunklet* and *chunk* ω values are computed the same was as *activity* ω values (Equation 3.3) but replacing ν for δ. For *user* u, and *chunklet or chunk* i, ω is computed as follows:

$$\omega_i = \frac{\rho_i}{\phi_i} \| \nu_i - \psi_u \odot \kappa_u \|_2 \tag{3.7}$$

Linear Optimization of chunks

After the first round of chunklet optimizations and attribute updates, Function 3 reiteratively calls a second linear optimization program (LP2) for each *chunk* in the subgraph of relevant content. LP2 evaluates the set of related *chunklet* nodes that have an *CONTAINS* relationship to each *chunk* in order to determine which of those *chunklets* should be recommended. The only different between LP1 and LP2 is the tier at which the LP evaluates content. Otherwise, LP1 and LP2 are computationally equivalent.

Linear Program 2

Sets and Indices:

$n \in N$ Set of related *chunklet* nodes

Data:

β_n	β value of n; (1=required by *chunk*)	$\beta \in \{0, 1\}$
ω_n	ω value of *chunklet* node n	$\omega \in \mathbb{R}^*$
υ_n	υ value of *chunklet* node n	$\upsilon \in \{0, 1\}$
γ_n	γ value of relationship n	$\gamma \in \mathbb{R}^*$

Decision Variables:

α_n Which *chunklet* nodes to recommend (indicated by $\alpha = 1$)

Objective Function:

$$\min_{\alpha} \sum_{n \in N} \alpha_n \omega_n \upsilon_n$$

Constraints:
$s.t.$

$$\sum_{n \in N} \alpha_n \gamma_n \geq 1 \qquad \text{(total completion credit of recommended chunklet must be } \geq 1\text{)}$$

$$\alpha_n - \beta_n \geq 0 \quad \forall n \in N \qquad \text{(if chunklet } n \text{ is required, } \alpha_n \text{ must = 1)}$$

$$\gamma, \omega \geq 0$$

$$\alpha, \beta, \upsilon \in \{0, 1\}$$

Once the solver has returned an optimal solution, the model's α values now indicated which chunklet modules are recommended for each chunk. The same α values are transferred back to their respective *CONTAINS* relationships in the same manner as after LP1.

Linear Optimization of Skills

The last sub-step of Function 3 is optimizing each skill in the set of required skills. The optimization of skills is performed similarly to the optimization of chunklets and chunks. Optimization of skills has the added complexity that skills can have multiple sets of choices rather than a single pool of sub-content to choose from (as seen in Figure 3.7). To account for these various sets in the third linear optimization program (LP3), we introduce an intermediary tier of nodes to the database graph called *sub-skills*.

Each *skill* is still directly related to relevant chunks through *TRAINED_BY* relationships. Any *chunk* nodes directly related to a *skill* without a set of choices is considered a required chunk and the associated relationship α and β values are set to 1. However, any set of choices (such as: "any two of three chunks") provided by the content authors or database administrators, requires computational interpretation through a *sub-skill* node. Each new sub-skill is related to the skill through a *REQUIRES* relationship, and to the associated *chunks* through *TRAINED_BY* relationships. The nature of the choice provided (i.e., "one of two", "three of five", etc.) will dictate the γ values for the *TRAINED_BY* relationship between the *sub-skill* and the *chunk* nodes.

Each *sub-skill* for each *skill* node is optimized as a constraint within LP3. LP3 optimizes content using the same algorithm as LP2 and LP1, with the difference being the tier at which

content is evaluated and the constraint that each *sub-skill*'s conditions must be satisfied. Also, the α values determined by LP3 are not stored on the relationship between the *chunks* and the *sub-skill*s, but only on the relationship between the *chunk* and *skill* nodes.

Linear Program 3

Sets and Indices:

$n \in N$	Set of adjacent *chunk* nodes
$i \in I$	Set of *sub-skill* nodes related to the evaluated *skill*
$s \in S_i$	Relationships between related relevant *chunk* nodes and given *sub-skill*$_i$

Data:

ω_n	ω value of *chunk* node n	$\omega \in \mathbb{R}^*$
υ_n	υ value of *chunk* node n	$\upsilon \in \{0, 1\}$
γ_s	γ value of relationship s	$\gamma \in \mathbb{R}^*$

Decision Variables:

α_n	$\forall n \in N$	Which *chunk* nodes to recommend (indicated by $\alpha = 1$)

Objective Function:

$$\min_{\alpha} \sum_{n \in N} \alpha_n \omega_n \upsilon_n$$

Constraints:

$s.t.$

$$\sum_{s \in S_i} \alpha_s \gamma_s \geq 1 \quad \forall i \in I \qquad \text{(total completion credit of chunks in each } \textit{sub-skill} \geq 1)$$

$$\alpha_n - \beta_n \geq 0 \quad \forall n \in N \qquad \text{(if chunk } n \text{ is required, } \alpha_n \text{ must = 1)}$$

$$\gamma, \omega \geq 0$$

$$\alpha, \beta, \upsilon \in \{0, 1\}$$

The output of LP3 is the set of relationship α values for the evaluated *skill*. These indicate which chunks are recommended content modules for the required skill.

3.5.4 Function 4: Build a Personalized Training Network

The desired recommender system output is a personalized training plan in the form of a network of recommended training content. Each personalized training network is built specifically for a single user based on that user's desired position(s). The network depicted in Figure 3.13 depicts a single user vertex directly connected (thick orange lines) to the chunks the Recommender System identifies as recommended content. This represents the content the user must complete in order to acquire all the skills required for the user's desired position. The user also has access to other related and relevant optional content (shown with thin dashed lines). These chunks also support the training on the absent necessary skills. However, these content choices will be a worse match for the user based on learning style and quality.

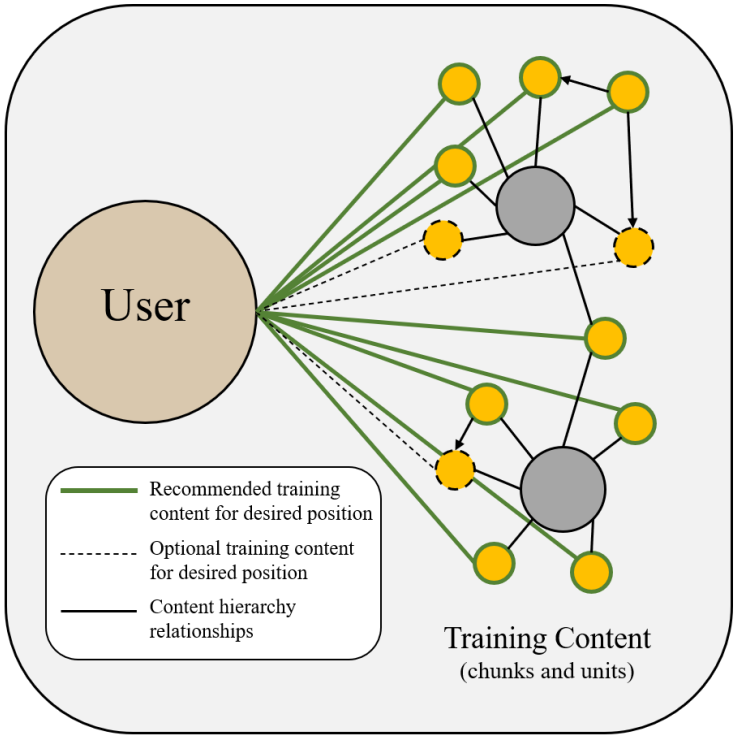


Figure 3.13. Personalized Training Network for a Generic User and Position

Once the linear optimizations in Function 3 converge on the optimal mix of content for each required skill, we need to convert those results into a new subgraph of recommended content. We repeat the BFS algorithm from Function 1, but this time we limit the algorithm

to only consider relationships that the recommender system has identified as recommended ($\alpha = 1$), and content which is incomplete ($\upsilon = 1$).

We begin by building recommender system links between the user-position combination and establish a *REC_SYS_LINK* relationship between them. Next, we add more *REC_SYS_LINK* relationships between the position and the skills that the user had not acquired. Then we add *REC_SYS_LINK* relationships between these skills and the recommended training content nodes in accordance with our BFS algorithm. At this point, the DBMS can produce a new subgraph of recommended content (shown in Figure 3.14) through a single CQL query such as:

```
MATCH (n)-[rel:REC_SYS_LINK]-(m) RETURN n,m,rel
```

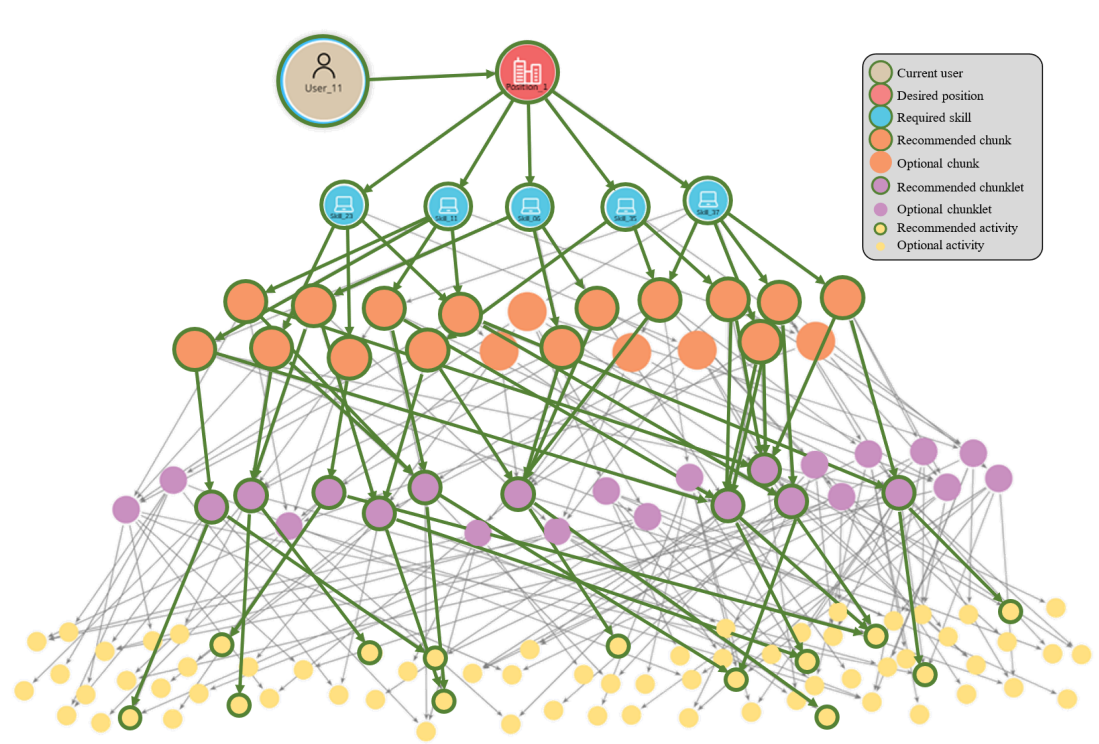


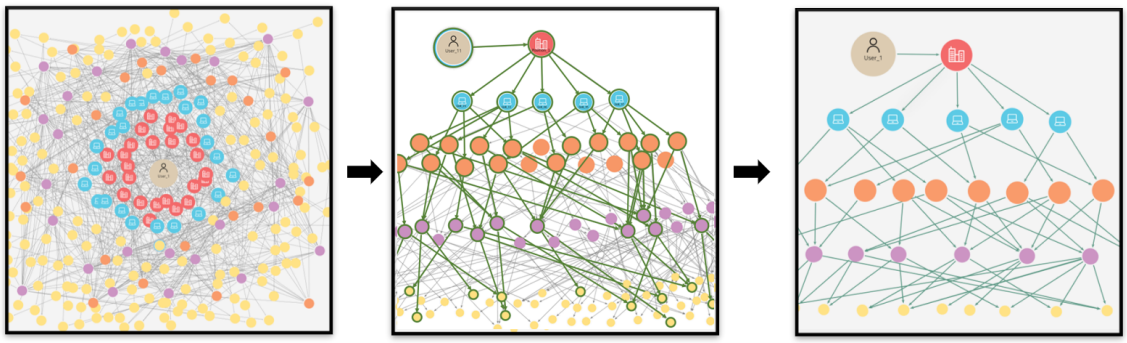
Figure 3.14. Personalized Training Network for a Specific User and Position

The personalized training network described above is presented to the user in a Graphical User Interface (GUI) which resembles the legacy CHUNK Learning Network (shown in Figure 2.7. Users interact with the network of training content through this interface in order

to acquire the skills required for the position(s) desired. At this point, a job-seeker should be able to clearly see that they can become qualified for the job they want by completing the recommended training activities (indicated by the yellow nodes with green circles at the bottom of the slide). Another feature of this environment which differentiates it from a traditional linear model is the system dynamically updates recommender system training plans. If a user becomes engaged with a particular set of activities or chunks, they can choose to conduct training on optional modules they find interesting. The system adjusts the personalized training network with each completed module to constantly display to current optimal training plan.

3.6 Methodology Recap

In summation, we develop this framework starting from a chaotic skills acquisition environment represented as a complex network. In the currently existing environment (shown in Figure 3.15[a]) a job-seeker may know what position they want, but they are surrounded and possibly overwhelmed by many professional skill development opportunities.



(a) Comprehensive training network

(b) Subgraph of relevant content with recommended content identified

(c) Personalized training network

Figure 3.15. CHUNK-Professional Development Process

In the newly created environment, namely CHUNK-PD, we organize and focus things for the job-seeker. We reform the network as a graph-database that stores the training content in an organized hierarchy, and we use a search algorithm to build a subgraph of the training content relevant to the job-seeker (shown in Figure 3.15[b]).

Next, we employ a hybrid recommender system which incorporates the job-seeker's learning style and the content map to identify optimal content to the job-seeker, in order to produce a training plan in the form of a personalized training network like the one seen in Figure 3.15(c) (or in Figure 3.14 in greater detail.)

This graph-database and hybrid recommender system combination enables us to personalize talent development utilizing a network framework. Personalized talent development which helps job-seekers identify the skills they need for the jobs they want and meaningful training with which to acquire those skills.

CHAPTER 4:
Framework Extensions and Future Research

In Chapter 3, we described the methodology for the proposed CHUNK-PD framework which addresses the challenges job seekers face in the personal knowledge and skill acquisition environment. In Chapter 4, we explore potential extensions which either expand the CHUNK-PD environment or enhance the framework's performance. We also propose some potential specific case usages where the CHUNK-PD framework would be well suited to addressing the challenges or needs of the case.

In Section 4.1, we introduce three extensions that can be applied to the framework which, while not required for the framework, enhance the effectiveness of the framework to match job-seekers to meaningful professional development. In Section 4.2, we suggest that CHUNK-PD would be of immediate benefit to a military application. Specifically, CHUNK-PD could be used by the Army Talent Management program to pursue its stated goals to win in a complex world. In Section 4.3, we change the paradigm and use CHUNK-PD to support organizational challenges rather than individual professional development. In Section 4.4, we discuss three areas where we believe the framework would benefit from additional research not covered in this work.

4.1 Extending the Model

The model presented in Chapter 3 is functionally complete without any of the following extensions, however, these extensions enhance the model's performance or they modify the model's algorithm to address additional considerations.

4.1.1 Testing and Skill Certification

While the responsibility for skill sufficiency remains with each individual job-seeker, the CHUNK-PD framework can incorporate testing and evaluation to certify new skill acquisition or to enhance user progress through the personalized training network. In this framework extension, users conduct a predetermined form of testing or evaluation after the completion of each chunk. The purpose of these evaluations is to demonstrate the job-

seeker's capacity to employ the associated skill being trained, therefore, evaluations are designed to emphasize skill proficiency over training material memorization. This is accomplished through "alternative assessments" that are tied to real-world complex problems and require judgement in the application of techniques [48]. "Alternative assessments" are complex and may have more than one set of possible solutions, therefore effective assessments require careful construction to be effective. Properly employed, these assessments provide a practical experiential event, allowing a job-seeker to demonstrate proficiency in multiple sub-topics simultaneously.

This CHUNK-PD extension employs two series of evaluations for each chunk, a pre-chunk test and a post-chunk test. The post-chunk testing certifies the user has sufficiently acquired the knowledge associated with the completed chunk. Passing a post-chunk test triggers a chunk completion in the personalized training network. Pre-chunk testing allows a user to demonstrate proficiency in a subject without having to complete the entire set of recommended training content activities and chunklets. The threshold for passing a pre-test is higher than a post-test since the job-seeker has not yet reviewed the professional development material.

Testing and evaluation content is derived from the required ($\beta = 1$) content that the job-seeker would have encountered while completing the recommended training plan. Both testing types adapt the personalized training plan to reflect the material that the job-seeker has not yet mastered. Therefore, if a job-seeker demonstrates proficiency in some areas of a post-chunk test, but fails to demonstrate proficiency in others, the testing process only recommends re-training and testing on material from chunklets the job-seeker has not mastered. Job-seekers are then free to review training material in those chunklets until they feel confident in their ability to demonstrate their capability to employ the associated skill once again. In some cases where a user fails a post-chunk test, the recommender system may adjust the personalized training plan to introduce new sources of training material; thus, giving the job-seeker an alternative method of learning the same skill related content.

By applying a series of evaluations to the CHUNK-PD environment which focuses on the job-seeker's capacity to employ a skill rather than an attributional grade, job-seekers are encouraged to focus their effort on professional development and not on course completion.

4.1.2 Matching Job-Seekers to Future Positions

The CHUNK-PD model is able evaluate a user's match to a specific position; as originally developed, this ability has the capacity to match a user to more than one position for training at a time. This capability allows a job-seeker to plan out a series of positions in a career progression, while finding professional development opportunities which support more than one position. This same capability allows a job-seeker to build a personalized training plan for multiple desired positions, using the recommender system to emphasize skills and training content that supports multiple positions for greater training efficiency. Without significant adjustment, we can extend the model to add a new recommender system that returns an ordered listing of "next best" jobs for a user based on minimized skill training and each user's personal interests and background. Positions can also be evaluated and compared based on physical locations, job requirements, or other attributes.

Alternatively, this principle could be reversed to approach the professional development problem from the organizational staffing perspective. The CHUNK-PD model conducts an automated matching evaluation of job-seekers to positions. If the matching evaluation is reversed, we can match a position to a pool of candidates, providing personalized training paths to candidates for that particular position. Using this extension, an organization which manages personnel can nominated employees for training or future positions based on the talents they already possess or anticipated organizational needs. While this CHUNK-PD extension would not directly address how an organization would identify and quantify future demands, it does nominate which employees to develop and provides a mechanism for them to train in an efficient manner.

4.1.3 Multilayered Composite Training Network

Recommender systems, like the one in the CHUNK-PD framework, become more effective as the quality of the input data increases. The CHUNK-PD framework was designed to effectively match job-seekers to professional development content with only a limited amount of data from the personal knowledge and skill acquisition environment. CHUNK-PD currently interacts with a single layered CTN, described in detail in Section 3.3. This extension proposes that the CHUNK-PD recommender system will produce more effective recommendation if it is modified to incorporate more data from a multilayered CTN.

This extension proposes that the CTN be expanded to include new layers of data for content, positions, and users that would contribute to the effectiveness of the recommender system. This data is captured on additional layers of the multilayered CTN. The first additional layer is the actors layer. This layer represents individuals or groups associated with training content or positions. Training content actor data could represent a specific training facilitator or course instructor. Position actor data could represent hiring representatives, management for the positions, or links to known co-workers of the position. While these may not seem to directly affect the professional development process for a job-seeker, they provide opportunities to build professional or mentorship relationships, or they provide an opportunity for a job-seeker to seek out individualized training from a training source they are already familiar with.

The next new layer is the time layer. This layer represents the time constraints that affect real-world professional development. The first and most concerning is training availability, highly matched training activities which are only available at certain times need to be scheduled. This temporal data is captured in this layer along with references to the job-seeker's availability (because we all have conflicting requirements on our time) for the recommender system to select and schedule activities that are only available at certain times. This same consideration will also prevent recommending training activities which only occur at overlapping times. This time layer also represents data on the positions such as when the position is available or hiring timelines. While the CHUNK-PD environment deliberately avoids using time duration as part of the content recommendation process, job-seekers can use this information to assist in planning their professional development.

The third proposed expansion layer is a spatial layer. In an ideal environment, every job-seeker would have virtual or local access to every professional development training artifact where they access the CHUNK-PD framework. However, some of the most interactive training activities may be associated with a physical location, such as an in-person event. Even if no such in-person training artifacts were recommended as part of training plan, job-seekers and positions are still associated with the physical locations where they reside. Accounting for distance requirements prevents recommending professional development to a job-seeker which they are unable or unwilling to travel to. Other training activities may be limited by the number of trainees who can attend. The data in this layer would provide a means for the recommender system to account for both of these challenges simultaneously.

This geographical data can also be crucial when comparing the matching between a user and multiple positions or a position to multiple job-seekers, as mentioned in Section 4.1.2

4.2 Military Application

The CHUNK-PD framework could assist the U.S. Army Human Resources Command manage the inherent talents of the 1.25 million personnel in the Army (including both Soldiers and Department of the Army (DA) civilians). For decades, the purpose of the Army's personnel divisions has been to allocate manpower according to end-strength needs. As the Army looks to maintain a competitive edge in the complex operating environment, it recognizes a need to better align the knowledge and skills of its human capital to better leverage the force it has now and will need in the future. The Army Talent Management strategy for 2025 describes a new personnel management approach that shifts the focus from distribution of manpower to "deliberately managing the talents that the Soldiers and civilians in the Army possess [49]."

The Army's Talent Management Concept of Operations describe talent management as an investment requiring a systems approach. CHUNK-PD would directly support this since CHUNK-PD would allow the Army's senior leaders to project strategy for future operations, then use the provided framework to identify personnel who already possess the talents and skills desired for emerging operations while selecting other Army personnel for professional development. In this manner the Army would be both investing in future operations while simultaneous transforming the force based on individual capabilities rather than raw manpower and Military Occupational Specialty (MOS) qualifications.

4.3 Organizational Level Professional Development

Considering the framework extensions suggested in Section 4.1 and the type of organizational demands alluded to in Section 4.2, the CHUNK-PD framework could be employed as a asset for managing organizational talent demands. In Section 4.2, we suggested the Army could use CHUNK-PD to promote individualized talent development and placement for specific service members. In this section we specifically employ CHUNK-PD to manage organizational talents and simultaneous professional development across every member of an organization.

Any organization which combines talents and skills from a team needs to balance work requirements with team member capabilities to successfully complete projects and tasks. In some settings this may be represented as a pool of employees in an office, where the office needs a set number of accountants, sales specialists, inventory managers, and customer service specialists on any given day. This could also be represented in a more technical project, an example project may require three members who are proficient in neural networks, four programmers, two graphic designers, and a project manager. CHUNK-PD works in both of these examples by maintaining the graph-database data for all available employees and recommending which combinations of employees best meets the requirements for special projects based on the skills they possess already.

What makes CHUNK-PD useful is the capability to support future organizational demands. In the previous example a human resources decision is made to select employees for current needs based on the human capital available. Effective organizations project demands for future organizational needs which usually do not match current human capital holdings.

An emerging project focusing on improving the artificial intelligence knowledge landscape within the federal government proposes a two-part organizational knowledge assessment and sustainment model in which organizations assess and sustain their workforce capabilities [50]. This model assesses knowledge requirements based on policy, technology, and external influences. When gaps are identified by the knowledge audit the organization must decide to either develop current employees, shift personnel from other positions, or hire new employees with the required talents. CHUNK-PD can support these type of organizational talent demand evaluations. Since CHUNK-PD maintains an active database of training content, positions, and employees, it can perform optimization evaluations to predict a (non-monetary) professional development cost for each employee to acquire new skills. These training costs can then be interpreted by the organization and its human resources systems to decide which employees to develop and when reorganization or hiring is a more appropriate choice.

This capability is valuable because it can consider entire organizations of employees interacting to meet talent or knowledge demands simultaneously. This capability supports but current and future organizational demands, preventing future organizational knowledge gaps.

4.4 Further Research

The most significant anticipated hurdle to implementation of the CHUNK-PD framework is the programmatic integration of positions and content into the network. New positions need to be manually pulled into the network and their meta data added by hand. New content that must also be manually added to the system and evaluated by the content authors and system administrators to determine the appropriate ELM delivery mode.

4.4.1 Programmatic Creation of Positions in the Network Model

Building and maintaining the collection of positions in the network currently requires significant manual data entry. Automating this process will improve the consistency of the position data and will enable to the network to dynamically update as new positions are generated and discovered. One potential method for automating this process includes developing a method of web scraping job posting websites for position data. Web scraping would be combined with a sequence of natural language processing to find required skills and attributes from job descriptions. As these new positions are discovered and the skills are identified, skills could be processed through a machine learning ensembles-based evaluation to categorize required skills or associate them with existing skills already in the framework.

4.4.2 Programmatic Evaluation of Content Training Mode

Building and maintaining the library of training content is the most significant hurdle to CHUNK-PD implementation. With further research, an automated method to incorporate training material into the content library could be developed. In the current framework, ELM content delivery modes are determined subjectively by content creators and system administrators. This is a time intensive process which is vulnerable to personal biases. An automated process would be more efficient and would add consistency to the assignment of ELM modes.

4.4.3 Asynchronous Interaction

Even though CHUNK-PD emphasizes personalized learning for job-seekers, there are many benefits associated with interpersonal interaction while developing new skills and completing training content. We suggest that future research should consider developing mechanisms to provide layers of interpersonal interaction in the asynchronous professional development

environment. These layers could be based on other job-seekers who are completing similar training activities at the same time, professional peers who may be in similar jobs to the job-seeker's desired position, or mentors who can guide the job-seeker with their experience, even if it is asynchronously.

In online education or virtual learning, students could be assigned to one or more groups based on who is enrolled in the same courses. These groups could be encouraged to meet virtually to study or complete homework together, as well as to share experiences that others can benefit from in similar situations. This peer interaction helps students find motivation to complete assignments and could enhance knowledge transfer by providing an additional source of assistance on the assigned academic material. Related research by Critchley [51] leverages a student's social network to recommend new content to explore, thus reinforcing material in their subject area or introducing them to new knowledge areas. His research reinforces the idea that strong social ties enhance knowledge transfer between students in the same field of interest, while weak social ties provide breadth in other areas of study.

These peer interaction mechanisms could be applied to the CHUNK-PD environment by identifying job-seekers who are training similar skills, on similar chunks or chunklets, and then assigning them to peer groups where they can socialize professional development topics and support each other. Similar to the ELM cycle that prescribes that learning is more effective when it incorporates multiple modes of learning, job-seekers may acquire skills more effectively if they have sympathetic peers who can assist in discussing the same content from a different perspective than the recommended training modules.

In addition to peers who are training on similar training content modules, job-seekers may benefit from social connections to employees in positions which require the skills the job-seekers are trying to acquire. By connecting to professionals who employ the skills in their current jobs, the job-seekers may gain insights for real-world applications and the latest techniques which may not have training content developed yet.

Finally, job-seekers should have access to meaningful guidance provided by a mentor. Research by Reeder [52] shows that effective mentorship encourages professional development through knowledge sharing. Reeder uses a network-based approach to identify prospective mentor-mentee relationships within the U.S. Navy based on similarities in assignments and personal attributes. What makes this approach useful is that is uses a compatible technique of

network-based matching and incorporates personal attributes that go deeper than assignment history. This same approach could be generalized to identify prospective mentor-mentee relationships across the CHUNK-PD environment where potential mentors and mentees with high similarities would be nominated. Once a mentor-mentee relationship is established in CHUNK-PD, the job-seeker can seek out long-term career advice and guidance from senior professionals who have experience using the same or very similar skills that the job-seeker is training to acquire and insights for successful long-term career development.

In summation the CHUNK-PD framework has additional potential through extensions, applications, and future research. The framework can be enhanced through extensions to provide a more accurate recommender system by incorporating "alternative assessments" which requires job-seekers to demonstrate judgement in the application of the training content material. The framework can also be expanded to incorporate additional layers of data to account for the people, time, and space associated with the professional development environment. CHUNK-PD can also be adapted to support organizational talent demands by evaluating the job matching values for a large workforce. In this capacity CHUNK-PD helps both users and jobs by identifying jobs for job-seekers and qualified candidates for the talents and skills in demand in an organization.

With further research the CHUNK-PD framework could also be enhanced for automated system growth and management while recapturing some of the benefits of personal connections. Professional development through CHUNK-PD was never intended to be a lonely process; overlaying social networks on top of the existing framework allows CHUNK-PD to identify peers, professionals, and mentors within the network who can support job-seekers in their professional development journey. With the development of programmatic systems, CHUNK-PD could automate the process of incorporating and evaluating new positions and content. This anticipated automation will allow the CHUNK-PD to digest and present training artifacts to job-seekers as soon as they are created and immediately enhance the library of training content for job-seekers.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions

In conclusion, the challenges associated with professional development and job-seeking remain complicated; but the CHUNK-PD environment provides a framework for matching job-seekers to training content. The CHUNK-PD network-based framework provides a mechanism for job-seekers to identify meaningful professional development for the skills they need for the jobs they want. Job-seekers benefit from personalized training plans of optimized content preparing them for future professional positions.

Job-seekers still face a crowded, chaotic, and complicated personal knowledge and skill acquisition environment. There are tens of millions of Job-seekers in the United States looking for a job today. In order to be competitive, these job-seekers need to identify and acquire the right skills for their desired positions. In an increasingly complicated world where technical skills are often closely related but still distinct enough to require separate training, finding meaningful content can be difficult and overwhelming. The variety of professional development opportunities also makes it difficult to find the content that best matches each job-seeker's needs and learning style.

The CHUNK-PD environment addresses both of the identified challenges job-seekers face (described in 1.1). It identifies the specific required skills for the desired position to include all the supporting training topics (known as chunks) directly related to each skill. CHUNK-PD also builds personalized training plans for each job-seeker which includes an optimized set of professional development content to guide the job-seeker to acquire the aforementioned skills in the most effective way possible.

CHUNK-PD utilizes network-science-based techniques such as search algorithms and graph-based databases to efficiently identify required skills and build personalized subgraphs of relevant training content for each job-seeker. Then, CHUNK-PD implements a hybrid recommender that combines collaborative-filtering and content-based systems to create personalized professional development curricula for each job-seeker. These curricula are built in the form of personalized training networks which allow the job-seekers to heuristically explore content as they prepare for future jobs.

The CHUNK-PD environment is designed to be effective without relying on burdensome initialization requirements from the job-seeker. Any job-seeker can easily establish a user profile in the environment and start building a training plan for any position known to the training network. Since it is a network-based framework with very little overhead data requirements, the entire CHUNK-PD environment is scalable and ready for cloud-based implementation for one or for millions of users. The network framework supported by a graph-database is also well-suited for efficient data transactions with a vast library of training content organized hierarchically.

The most important feature of the CHUNK-PD framework is that the recommender system mimics human behavior and selects training activities that will appeal to the job-seeker's learning style while dynamically adjusting the training plan as job-seekers explore professional development content. This feature promotes skill acquisition through personalized learning that results in job-seekers who are better equipped with talents for their desired positions.

CHUNK-PD is also an adaptable framework. With the addition of testing and skill certification, job-seekers and potential employers gain increased confidence in the skills that job-seekers have acquired. The same professional development framework can also be used to match positions and job-seekers based on the similarity of the skills and attributes they already possess. This framework has a direct application for organizations with significant human capital requirements. CHUNK-PD could support the Army's Talent Management Strategy but evaluating the talents and skills of the entire force and match service members to training content to grow the pool of available talent in projected areas of future demand.

Finding and getting qualified for the job you want can be hard, but the CHUNK-PD framework can make it easier to identify the skills required for the job you want, and recommend a meaningful professional development training plan to guide you to acquiring those skills.

List of References

[1] Bureau of Labor Statistics, "Employee tenure in 2020," *Bureau of Labor Statistics*, Sep 2020. Available: https://www.bls.gov/news.release/pdf/tenure.pdf

[2] Pew Research Center, "The state of American jobs," *Pew Research Center*, October 2016. Available: https://www.pewresearch.org/social-trends/2016/10/06/the-state-of-american-jobs/

[3] S. Oh, D. M. DiNitto, and D. A. Powers, "A longitudinal evaluation of government-sponsored job skills training and basic employment services among U.S. baby boomers with economic disadvantages," *Evaluation and Program Planning*, vol. 82, p. 101845, 2020.

[4] L. F. Katz, "America's jobs challenges and the continuing role of the U.S. Department of Labor," *ILR Review*, vol. 67, no. 3_suppl, pp. 578–583, 2014.

[5] U.S. Bureau of Labor Statistics, "Occupational outlook handbook," 2020. Available: https://www.bls.gov/ooh/home.htm

[6] OECD, *Getting skills right: Assessing and anticipating changing skill needs*. OECD, 2016. Available: https://www.oecd-ilibrary.org/content/publication/9789264252073-en

[7] L. Euler, "Leonhard Euler and the Königsberg Bridges," *Scientific American*, vol. 189, no. 1, pp. 66–72, 1953.

[8] Chartrand, Gary and Zhang, Ping, *A First Course in Graph Theory*. Mineola, NY: Dover, 2013.

[9] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[10] Albert-Lásló Barabási. (2016, August). *Network Science*. Cambridge University Press [online]. Available: http://networksciencebook.com/

[11] R. Gera, "Betweenness Centrality," class lecture for Complex Networks, Dept. of Applied Mathematics, Naval Postgraduate School, Monterey, CA, 2021. Available: http://faculty.nps.edu/rgera/MA4404/06-CentralitiesBetweenness.pptx

[12] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[13] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'Small-World' Networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[14] Q. K. Telesford, K. E. Joyce, S. Hayasaka, J. H. Burdette, and P. J. Laurienti, "The ubiquity of Small-World Networks," *Brain Connectivity*, vol. 1, no. 5, pp. 367–375, 2011.

[15] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[16] R. Gera, "Community Detection," class lecture for Complex Networks. Available: http://faculty.nps.edu/rgera/MA4404/06-CentralitiesBetweenness.pptx

[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Upper Saddle River, NJ: Prentice Hall, 1993.

[18] M. Gyssens, J. Paredaens, J. Van den Bussche, and D. Van Gucht, "A graph-oriented object database model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 4, pp. 572–586, 1994.

[19] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 26, no. 1, pp. 64–69, 1983.

[20] H. Huang and Z. Dong, "Research on architecture and query performance based on distributed graph database Neo4j," in *2013 3rd International Conference on Consumer Electronics, Communications and Networks*. IEEE, 2013, pp. 533–536.

[21] D. Fernandes and J. Bernardino, "Graph databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB," in *DATA*, 2018, pp. 373–380.

[22] R. K. Sawyer, *The Cambridge Handbook of the Learning Sciences*. Cambridge University Press, 2005.

[23] J. Dewey, "Experience and education," in *The Educational Forum*, no. 3. Taylor & Francis, 1986, vol. 50, pp. 241–252.

[24] J. F. Pane, E. D. Steiner, M. D. Baird, and L. S. Hamilton, "Continued progress: promising evidence on personalized learning." *Rand Corporation*, 2015.

[25] P. Grant and D. Basye, *Personalized Learning: A Guide for Engaging Students with Technology*. International Society for Technology in Education, 2014.

[26] S. Sinek, *Start With Why: How Great Leaders Inspire Everyone to Take Action*. Penguin, 2009.

[27] J. W. Gentry, "What is experiential learning," *Guide to Business Gaming and Experiential Learning*, vol. 9, p. 20, 1990.

[28] J. D. Hoover and C. J. Whitehead, "An experiential-cognitive methodology in the first course in management: Some preliminary results," in *Developments in Business Simulation and Experiential Learning: Proceedings of the Annual ABSEL Conference*, 1975, vol. 2.

[29] D. A. Kolb, *Experiential learning: Experience as the Source of Learning and Development*. FT Press, 1984.

[30] D. A. Kolb, *The Kolb Learning Style Inventory*. Hay Resources Direct Boston, MA, 2011.

[31] A. Y. Kolb and D. A. Kolb, "Learning styles and learning spaces: Enhancing experiential learning in higher education," *Academy of Management Learning & Education*, vol. 4, no. 2, pp. 193–212, 2005.

[32] N. A. Sahabudin and M. B. Ali, "Personalized learning and learning style among upper secondary school students," *Procedia-Social and Behavioral Sciences*, vol. 103, pp. 710–716, 2013.

[33] Gera, Ralucca and Bartolf, D'Marie and Isenhour, Michelle L and Tick, Simona, "CHUNK: curated heuristic using a network of knowledge," *The Fifth International Conference on Human and Social Analytics. HUSO*, 2019.

[34] J.-D. Cleven, "A multilayer network approach for real-time adaptive personalized learning," M.S. thesis, Naval Postgraduate School, Monterey, CA, 2018.

[35] R. Johnson and D. Bartolf, "About CHUNK learning," NPS Wiki, November 2020. Available: https://wiki.nps.edu/display/CHUNKL/About+CHUNK+Learning

[36] F. Ricci, L. Rokach, and B. Shapira, "Recommender Systems Handbook," in *Recommender Systems Handbook*. New York, NY, USA: Springer, 2011.

[37] K. M. Benzi, "From Recommender Systems to Spatio-Temporal Dynamics with Network Science," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2017.

[38] J. S. Underwood, "Metis: A content map-based recommender system for digital learning activities," in *Educational Recommender Systems and Technologies: Practices and Challenges*. Hershey, PA: IGI Global, 2012, pp. 24–42.

[39] D. Wu, J. Lu, and G. Zhang, "A fuzzy tree matching-based personalized e-learning recommender system," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2412–2426, 2015.

[40] U.S. Government, "Operations research analyst," USA Jobs, February 2021. Available: https://www.usajobs.gov/GetJob/ViewDetails/591067000

[41] Neo4j, "The Neo4j Graph Data Platform," 2021. Available: https://neo4j.com/product/

[42] Neo4j, "Cypher Query Language," 2021. Available: https://neo4j.com/developer/cypher/

[43] Brown, Gerald G and Dell, Robert F, "Formulating integer linear programs: A rogues' gallery," *INFORMS Transactions on Education*, vol. 7, no. 2, pp. 153–159, 2007.

[44] R. L. Rardin, *Optimization in Operations Research*, 2nd ed. New York, NY, USA: Pearson, 2017.

[45] J. J. Forrest, S. Vigerske, H. G. Santos, T. Ralphs, L. Hafer, B. Kristjansson, J. P. Fasano, E. Straver, M. Lubin, R. Lougee, J. P. Goncal, H. I. Gassmann, and M. Saltzman, *COIN-OR / CBC: Version 2.10.5*, Mar 2020. Available: https://doi.org/10.5281/zenodo.3700700

[46] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: Modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[47] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-Optimization Modeling in Python*, 2nd ed. Springer Science & Business Media, 2017, vol. 67.

[48] P. Rousseau, "Best practices in alternative assessments," Ryerson University: Teaching and Learning Office, Toronto, 2018. Available: https://www.ryerson.ca/content/dam/learning-teaching/teaching-resources/assessment/alternative-assessments.pdf

[49] *U.S. Army Talent Management Strategy*, Department of the Army, Washington D.C., 2016. Available: https://talent.army.mil

[50] C. Manuel, D. Bartolf, M. Schlesinger, and R. Gera, "Teaching and Education Assessment Marketplace Brief," Dept. Graduate Education, Naval Postgraduate School (NPS), Monterey, CA, 2021, unpublished.

[51] M. Critchley, "A recommender model using social tie strength for the CHUNK Learning system," M.S. thesis, NPS, Monterey, CA, 2021.

[52] T. Reeder, "Analysis of a similarity-based approach to positively affect mentor-mentee relationships among service members," M.S. thesis, NPS, Monterey, CA, 2021.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California